Bauhaus-Universität Weimar

Data coupled civil engineering applications: Modeling and quality assessment methods

(Datenkopplung für Anwendungen im Bauingenieurwesen: Methoden zur Modellierung und Qualitätsbewertung)

DISSERTATION

Zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

an der Fakultät Bauingenieurwesen
der
Bauhaus Universität Weimar

vorgelegt von

Dipl.-Ing. Toni Fröbel
geboren am 11. März 1977 in Cottbus

Mentor: Dr.-Ing. Berthold Firmenich,
Junior-Professor an der Bauhaus-Universität Weimar von 2002 bis 2010
Mitberichter: Prof. Dr.-Ing. habil. Carsten Könke, Bauhaus-Universität Weimar
Mitberichter: Prof. Robert Amor, The University of Auckland, New Zealand

Eingereicht am: 13. Dezember 2011
Tag der Disputation: 20. August 2012

Impressum

Schriftenreihe des DFG-Graduiertenkollegs Modellqualitäten, Heft 6
Herausgegeben von der Fakultät Bauingenieurwesen der Bauhaus-Universität Weimar

Autor: Toni Fröbel
Data coupled civil engineering applications: Modeling and quality assessment methods
(Datenkopplung für Anwendungen im Bauingenieurwesen: Methoden zur Modellierung und Qualitätsbewertung)

Umschlag: Antje Danz
Druck: docupoint Magdeburg GmbH
Verlag der Bauhaus-Universität Weimar 2013

ISBN: 978-3-86068-486-3

Bestellungen:
verlag@uni-weimar.de
Fax: 03643/581156

Abstract

The planning process in civil engineering is highly complex and not manageable in its entirety. The state of the art decomposes complex tasks into smaller, manageable sub-tasks. Due to the close interrelatedness of the sub-tasks, it is essential to couple them. However, from a software engineering point of view, this is quite challenging to do because of the numerous incompatible software applications on the market. This study is concerned with two main objectives: The first is the generic formulation of coupling strategies in order to support engineers in the implementation and selection of adequate coupling strategies. This has been achieved by the use of a coupling pattern language combined with a four-layered, metamodel architecture, whose applicability has been performed on a real coupling scenario. The second one is the quality assessment of coupled software. This has been developed based on the evaluated schema mapping. This approach has been described using mathematical expressions derived from the set theory and graph theory by taking the various mapping patterns into account. Moreover, the coupling quality has been evaluated within the formalization process by considering the uncertainties that arise during mapping and has resulted in global quality values, which can be used by the user to assess the exchange. Finally, the applicability of the proposed approach has been shown using an engineering case study.

Keywords:
Data exchange, Schema mapping, Quality assessment, Uncertainty, Coupling, BIM, Design patterns, Metamodel architecture

Zusammenfassung

Der Planungsprozess im Bauwesen ist hochkomplex und daher in seiner Gesamtheit nicht zu erfassen. Deshalb wird dieser in kleinere und beherrschbarere Teilaufgaben zerlegt. Auf Grund ihrer starken Wechselwirkungen ist deren Kopplung unabdingbar. Aus Sicht der Informatik wird dies jedoch durch eine große Anzahl inkompatibler Softwareanwendungen erschwert. Die Arbeit beschäftigt sich daher mit zwei wesentlichen Aufgabenfeldern im Bereich der Softwarekopplung. Als erstes werden Kopplungskonzepte unabhängig von spezifischen Hardware- oder Softwareeigenschaften beschrieben, um den Ingenieur bei der Durchführung und Auswahl von entsprechenden Kopplungsstrategien zu unterstützen. Dies wird durch eine Kopplungs-Mustersprache in Verbindung mit einer Meta-Modell-Architektur erreicht. Seine Anwendbarkeit wird an einem Kopplungsszenario gezeigt. Das zweite Aufgabenfeld beschäftigt sich mit der Qualität von gekoppelten Softwaresystemen. Eine Qualitätsbewertung erfolgt hierbei auf Basis von bewertetem Schema-Mapping. Der Ansatz ist auf Grundlage der Mengen- und Graphentheorie mathematisch beschrieben. Er berücksichtigt die gängigen Mapping-Muster und Unsicherheiten, die während des Mappingprozesses auftreten können. Der Bewertungsprozess liefert einen globalen Qualitätswert, der vom Ingenieur direkt verwendet werden kann, um den Austausch zu bewerten. Die Anwendbarkeit wird an einem Beispiel gezeigt.

Schlagwörter:
Datenaustausch, Schema-Mapping, Qualitätsbewertung, Unsicherheiten, Kopplung, BIM, Entwurfsmuster, Meta-Modell-Architektur

Vorwort

Die vorliegende Arbeit entstand in den Jahren 2008 bis 2011 während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Graduiertenkolleg 1462 *"Bewertung gekoppelter numerischer Partialmodelle im Konstruktiven Ingenieurbau"* an der Bauhaus-Universität Weimar.

Ich möchte mich an dieser Stelle bei all jenen Personen bedanken, die zum Gelingen dieser Arbeit beigetragen haben.

Mein besonderer Dank gilt meinem Mentor, Herrn Dr.-Ing. Berthold Firmenich, für die Anregung und Förderung dieser Arbeit sowie für die ausgezeichnete Betreuung. Besonders wertvoll waren die vielen fachlichen Diskussionen, das entgegengebrachte Vertrauen und seine Offenheit als Rahmen für diese Arbeit. Bei Herrn Prof. Dr.-Ing. habil. Carsten Könke und Herrn Prof. Dr. Robert Amor bedanke ich mich für die Begutachtung dieser Arbeit und für die offenen und zielführenden Diskussionen zum Thema. Ich habe mich sehr darüber gefreut, dass sich beide sofort bereit erklärt haben meine Arbeit zu begutachten, und Prof. Amor den langen Weg aus Neuseeland nicht gescheut hat, um meiner Disputation in Weimar beizuwohnen.

All meinen Kolleginnen und Kollegen danke ich ganz herzlich für die angenehme Zusammenarbeit und für die freundschaftliche Arbeitsatmosphäre, die vielen anregenden Gespräche und Diskussionen. Es war mir eine große Freude, in diesem Graduiertenkolleg zu arbeiten und Freunde fürs Leben gefunden zu haben.

Ich danke insbesondere der Deutschen Forschungsgemeinschaft (DFG), die mir die Möglichkeit der Promotion im Rahmen des Graduiertenkollegs 1462 ermöglicht hat.

Schließlich geht das größte Dankeschön an meine Familie, insbesondere an meine Eltern, Karin und Thomas. Sie haben mich stets auf meinem Lebensweg unterstützt, mir die Möglichkeit zum Studium gegeben und waren mir ein Ruhepol in schwierigen Phasen der Promotion.

Toni Fröbel

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Außer meiner waren keine weiteren Personen an der inhaltlich-materiellen Erstellung der vorliegenden Arbeit beteiligt. Insbesondere habe ich hierfür nicht die entgeltliche Hilfe von Vermittlungs- bzw. Beratungsdiensten (Promotionsberater oder anderer Personen) in Anspruch genommen. Niemand hat von mir unmittelbar oder mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Ich versichere ehrenwörtlich, dass ich nach bestem Wissen die reine Wahrheit gesagt und nichts verschwiegen habe.

Weimar, Dezember 2011

Toni Fröbel

Contents

1 Introduction

The planning process of buildings is highly complex and not manageable in its entirety. The state of the art decomposes complex tasks into smaller, manageable sub-tasks, which can then be solved separately and concurrently by different engineers with the aid of more or less specialized software applications. Due to the close interrelatedness of sub-tasks it is essential to couple them. From a software engineering point of view, this is challenging because the numerous software applications on the market are heterogeneous.

This thesis is strongly related to this problem domain. The subject of this paper was developed from a sub-project of the Research Training Group 1462[1], which was funded by the German Research Foundation.[2] The overall goal of the research group is to build up a methodical basis that can assure the quality of prognosis models in structural engineering in a quantitative manner [3]. In order to do this, various sub-tasks such as material behavior, structural response, environmental conditions must be considered. Due to the large number of different sub-tasks, the research group has been divided into twelve sub-projects. Each sub-project was assigned to one of the following four research topics: *theoretical basics*, *material behavior*, *model couplings*, or *cooperation platforms*. These four areas focus on sub-problems in the domain of response behavior of engineering structures, quality prediction of material behavior, interaction between models and software coupling. The sub-tasks are handled as follows: first, they are based on mathematical formulations, which can be attributed as partial models. In order to support the engineer, the partial models then have to be formulated on a computational basis, which for the most part, is embedded in complex software applications. However, partial models are interrelated. Therefore, the coupling of these partial models as shown in Figure 1.1 became an important task in the research group in order to achieve the overall goal and to deal with complex engineering questions. Coupling has to be carried out for both the mathematical and computational descriptions of the partial models.

[1] GRK 1462 – Evaluation of Coupled Numerical Partial Models in Structural Engineering.
[2] DFG – Deutche Forschungsgemeinschaft.

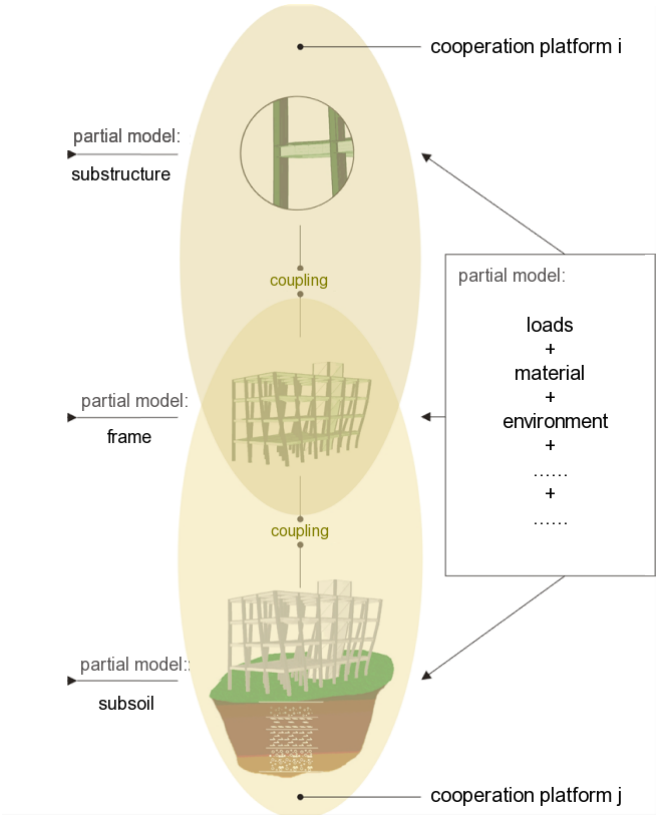


Figure 1.1: Coupling of partial models.

1.1 Statement of the Problem

This thesis is related to the research topic of *cooperation platforms*. It investigates the coupling of software applications and the embedded partial models to enable cooperative work within the research group to accomplish the overall goal. However, from a software engineering point of view, this is challenging to do because of the numerous incompatible software applications on the market. This study is concerned with two main objectives: The first is the generic formulation of coupling strategies. The second one is the quality assessment of coupled software.

- The coupling of software applications is challenging because finding a coupling solution is made complicated due to heterogeneous software environments. Software applications use their own data structures, which have been optimized to solve specific tasks and may run on different operating systems and computers. Furthermore, they are implemented by different programming languages. These languages are also based

on different programming paradigms. Furthermore, software development is a continuous process that is influenced by the rapid evolution in computer science. Software technologies and paradigms are generally developed in short cycles; and are used and adopted by a software to improve and extend its functionality. Thus, various coupling strategies are needed, and also have to be adjusted to these new software concepts.

- In addition, the design material in engineering practice (civil, mechanical and industrial engineering) is very sensitive. Lost or incorrect data can lead to numerous problems. Therefore, an error-free exchange of digital data is one of the most important factors to enable collaboration and ensure that work is properly distributed across disciplines and organizations. One widespread strategy in civil engineering is data structure coupling, which is when software applications are coupled on the basis of their internal data structures. Therefore, several data mappings between the used data structures are required. Due to incompatible data schemas and the many data mappings, perfect semantic interoperability cannot be expected. The quality of the coupling depends on the quantity and accuracy of data transferred. Hence, a further challenge is assessing the quality of the coupling between software applications.

1.2 Scope and Research Objectives

The research objectives of this thesis are related to the two main challenges mentioned in the previous section. The first part deals with the coupling of software applications and the second part is concerned with the quality assessment of couplings. Software coupling in computer science refers to making it possible for software applications to work together. In order to achieve this, two main issues have to be addressed. First, the software must be able to communicate with each other. Secondly, the exchanged data must be understandable. However, this is challenging because there are numerous data models and communication technologies available. This fact inevitably results in different coupling strategies. The capabilities of these various coupled software systems range from simple file exchange to full integration of information and processes across all stages of a construction project's life cycle. Furthermore, communication technologies are mostly restricted with respect to programming languages and operating systems used. As a result, implementations of coupling concepts are mostly limited to their coupling scenarios, which means they cannot be reused for other scenarios. Therefore, one of the objectives of this research was to develop a generic description of reusable coupling strategies according to the following considerations:

- The description should be independent of specific software properties.
- The coupling concepts should be classified in line with essential coupling aspects, such as runtime behavior or the direction of information flow.
- Finally, in order to prove the applicability of the description, the generic formulation should be performed in a real coupling scenario.

Quality assessment in computer science is classically achieved by software metrics. They are a measure of how well a software or its specifications have been designed and implemented by taking measurements into account that are objective, reproducible and quantifiable, such as budget and schedule planning, cost estimation, software structure, etc. However, the quality of coupled software systems primarily depends on the quantity and accuracy of the data to be transferred. Therefore, classic software metrics are not suitable for performing an assessment of coupled software. Nevertheless, current processes for evaluating data exchange can be used in order to achieve such an assessment of coupled systems. They mainly operate on data and work a posteriori. Changes resulting from inadequate data mappings are identified either by visual inspection or file comparison. Visual inspection refers to the detection of changes in data directly on the screen, or on printed drawings, which is carried out by the engineer. It is evident that visual inspection can only detect major problems, like missing or misrepresented elements. The file comparison approach can detect further minor changes in data, but it is limited. The data exchange must obey a common data format, which should preferably be a standard. In addition, objects have to be unique and distinguishable from one another. Furthermore, the entire process of mapping and file comparison is time and resource consuming and the results have to be qualitatively evaluated by the user. Therefore, in order to address the research objective that arises from this, an approach for assessing coupling quality based on a data schema rather than on its instances was developed, which takes the following into consideration:

- This approach is formalized mathematically by using the set theory and graph theory and by taking the various mapping patterns into account.
- Moreover, the coupling quality is evaluated within the formalization process, which takes the uncertainties arising from the mapping process into account. The resulting global quality values can be used by the user to assess the exchange.
- Finally, in order to prove the applicability of the proposed approach, an engineering case study is used.

1.3 Outline of Thesis

The thesis is structured as follows:

Chapter 2 is concerned with the state of the art of the essential methods and concepts needed within the context of this thesis. It includes methods for modeling and developing software, discusses existing coupling classifications, introduces concepts for achieving interoperability between software systems and examines approaches in order to assess certain quality aspects.

Chapters 3–5 outline how software applications can be coupled. In particular, chapter 3 investigates coupling aspects that are important factors in software coupling. They are the nodes in a self-developed decision tree. The coupling aspects support the development and implementation of coupling strategies. In addition, they also help in selecting adequate coupling strategies from existing ones Chapter 4 includes a coupling pattern language consisting of coupling templates and its instances. They are assigned to the nodes of the decision tree and provide reusable knowledge to achieve software coupling. Chapter 5 contains an example of a coupling scenario.

Chapters 6–8 discuss the coupling quality of software applications. In particular, chapter 6 introduces an approach to evaluating schema mapping in order to assess coupling quality on the basis of data exchange. The approach is formalized by using the set theory and graph theory and by taking the various mapping patterns into account. Uncertainties arising within the mapping process are considered as well. Chapter 7 includes the basic principles necessary for performing a schema analysis. They are essential for schema mapping due to the fact that data structures inside of a data schema may be interrelated to describe more complex facts. The principles takes certain types of relationships with respect to the object-oriented programming paradigm into account. Chapter 8 contains a case study for assessing coupling quality using one of the reference objects of the research group.

Chapter 9 contains the summary and outlook.

2 State of the Art

2.1 Software Modeling and Design

Software plays an essential role in the problem solving of complex tasks and is widely applied in the domain of civil engineering. However, software development is challenging with respect to the numerous programming languages and paradigms available. This fact is an important aspect of this thesis since the coupling of software is also a piece of software. In software development, programming paradigms define the fundamental ways of how software can be implemented. They differ with respect to the concepts and levels of abstraction for implementing the elements of a program, e.g., objects, methods, attributes, constraints, relationships, etc. Programming languages can support one or multiple programming paradigms. An overview of multi-paradigm programming languages is given in [73]. A classification of programming languages is shown in Figure 2.1.

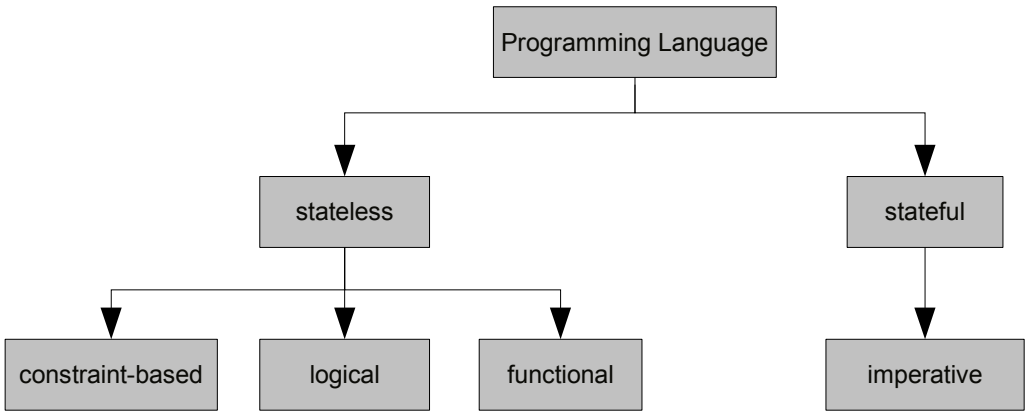


Figure 2.1: Classification of programming languages by Grabmüller [73].

One of the most popular programming paradigms is the object-oriented paradigm adopted by the group of stateful, imperative programming languages. It is widely used for software development within the domain of civil engineering, therefore, the focus of this thesis is the object-oriented paradigm. The basis of object-oriented programming (OOP) is provided by

objects. They can be understood as data structures consisting of data (attributes), behaviors (methods) and their interactions. The data within objects are not directly accessible, instead, they are accessed by special methods in order to ensure data consistency. Objects are capable of receiving messages, processing data, and sending messages in order to enable distinct roles or responsibilities. Available concepts and techniques for designing software on different levels of abstraction are data abstraction, encapsulation, messaging, modularity, polymorphism, and inheritance. The description and instantiation of objects occurs via classes, which can be understood as templates for creating objects. Classes can be interrelated on an attribute or class level. However, due to the numerous object-oriented programming languages available, it is important to consider software analysis and design in the implementation of such software systems and their couplings.

Software design is the process of finding the right software solution with respect to certain criteria, such as compatibility, extensibility, fault-tolerance, reliability, reusability, robustness, etc. It has to take the various design concepts into account, such as abstraction, refinement, modularity, data structure and information hiding. With regard to the object-oriented paradigm, this process is known as object-oriented analysis and design (OOAD).

- Object-oriented analysis (OOA) focuses on the problem domain and *what* the system does. It leads to a conceptual model, which does not consider the specific implementation details and constraints.

- Object-oriented design (OOD) focuses on the implementation details and on *how* the system does it. A system design is developed based on the conceptual model. It includes the implementation details and constraints of the specific programming languages and system architectures.

Due to the numerous object-oriented programming languages and many levels of abstraction, various modeling languages have been developed for certain domains, such as data modeling, process modeling and systems modeling. They are independent of a specific programming language and may be graphical or text-based. The modeling languages used for this thesis are the Unified Modeling Language (UML) (chapter 7) and Design Patterns (chapters 3 and 4). Furthermore, the modeling languages can support one another. As an example, design patterns uses the UML to describe the software architectures of problem solutions independently from the programming languages.

2.1.1 Unified Modeling Language (UML)

The Unified Modeling Language is a standardized, graphical-based modeling language in the domain of object-oriented software engineering. Its history can be traced back to the 1990s. UML was developed by the Object Management Group (OMG) [76] and is defined by a four-layered meta-modeling architecture, called the Meta-Object Facility (MOF), which is mentioned in subsection 3.3.1. The UML includes a set of graphical notations and techniques for data modeling, business modeling, object modeling and component modeling in order to cover all the processes within the software development life cycle. It defines thirteen types of diagrams as shown in Figure 2.2, which can be assigned to one of the following three categories: *structure diagrams*, *behavior diagrams* and *interaction diagrams*.

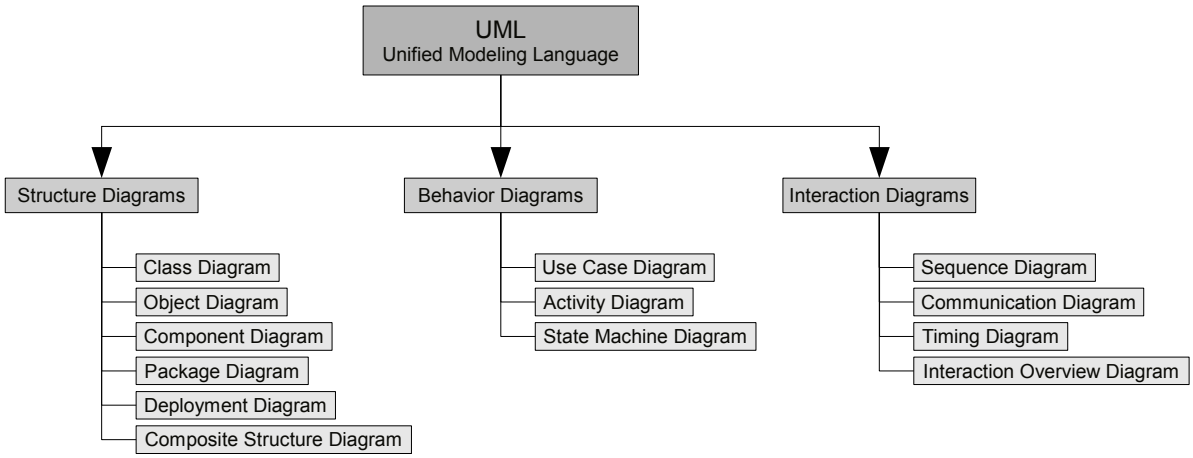


Figure 2.2: The types of diagrams defined by the UML.

Structural views on software systems, including their objects, attributes, methods and relationships are visualized through class diagrams and composite structure diagrams. A class diagram describes the architecture of a software system through graphical-based notations of the system's classes, their attributes, and the relationships among the classes. Class diagrams are widely used for data modeling in software engineering in order to create data schemas. The process of data modeling not only defines pure data elements, but the structures and relationships between them as well. Data schemas are essential within the context of this thesis for computing the quality of software systems with data-structured coupling (chapter 6). For this reason, the proposed generic approach for assessing coupling quality on the basis of data schemas is adapted to the object-oriented programming paradigm and its data modeling techniques (chapter 7). It includes the different types of relationships, such as general relationships, instance-level relationships and class-level relationships.

2.1.2 Design Patterns

Design patterns are used for solving problems in a wide domain of applications, which have became an inherent part of computer science. In general, a design pattern can be understood as a guide for making a design decision rather than providing a concrete design solution. It describes the reusable solution/guideline for a design problem in a formal way. For this purpose, it is structured into sections. Each section covers a specific issue, such as why a situation causes problems, in what situation the solution is applicable, or how the participants should collaborate. Design patterns were initialy used in the domain of architecture [8] and was later successfully adapted to the domain of computer science. Furthermore, a design pattern can be collected to form a pattern language, which then allows a common terminology to be defined. However, a design pattern is more or less a formal abstract description of a solution, therefore, it cannot be transformed directly into code.

Patterns in Computer Science

The concepts of patterns and pattern languages [37] are an inherent part of software design and architecture, however, they are also quite common in other aspects of software as well, such as databases and in HCI[1]. They are widely used to provide reusable solutions for frequently occurring problems. Solutions are often described as a combination of plain text, source code snippets and graphical UML elements. Patterns can be classified according to the level on which they operate, as follows:

- Design patterns are concerned with the technical aspects of an implementation. They provide solutions for problems arising from software design. Design patterns operate on the specific parts of the system, such as modules and interconnections. Thus, they only have a local impact on the implementation of a system.

- Architectural patterns target the strategic aspects of an implementation. They provide solutions for problems arising from the software architecture. In contrast to the design patterns, the architectural patterns operate on a higher level to solve or delineate the essential cohesive elements of a software architecture. Thus, they have a global impact on the overall implementation of a system.

The first design patterns in computer science were introduced in 1987 and used for designing graphical windows in Smalltalk[2] [28]. However, the breakthrough of design patterns in the

[1] Human–Computer Interaction, sometimes referred to as Man–Machine or Computer–Human Interaction.
[2] Object-oriented programming language.

domain of computer science took place in 1994 with publication of the book *Design Patterns: Elements of Reusable Object-Oriented Software*[67] written by the Gang of Four[3]. It consists of 23 classic patterns, classified into three categories. Implementations of the GoF patterns are available for different programming languages [42, 74, 116, 117].

Since the breakthrough, many patterns (especially architectural patterns) have been developed to cover various aspects of software design and architecture. A system of general patterns used mainly for structural decomposition, distributed systems and resource management can be found in [36, 38]. Patterns for concurrent and networked systems, e.g., service access, event handling, synchronization and concurrency, are available in [36, 155]. Patterns for resource management according to their acquisition, life cycle and release are introduced in [36, 99]. Patterns in the domain of distributed computing, which are related to resource managment, adaptation and extension, distribution infrastructure and interface partitioning are summarized in [36]. Patterns covering security aspects on different levels are collected in [156]. Patterns for implementing internet systems according to their performance, control and evolution can be found in [44]. Patterns for server components, including core infrastructure, component building blocks, component environment and deployment, as well as their implementation for different component-based architectures are addressed in [174]. There are hundreds of patterns regarding challenging problems in software engineering, such as remoting patterns [173], computer-mediated interaction patterns [157], patterns for error handling [78], and patterns for enterprise information applications [58, 82]. An excellent overview of documented and published software patterns is given in [150].

However, coupling strategies for software applications have to consider various semantic and technical decisions (subsection 3.1) and problem solutions (subsection 3.2). Unfortunately, no pattern language or patterns exist for coupling software applications. It is clear that some patterns may solve technical problems, such as synchronization, concurrency, or distributed working, however, these only provide partial solutions, which are limited to the overall coupling process. Consequently, a coupling pattern language is introduced (chapter 4) in order to define a common coupling language. It consists of a collection of coupling templates (subsection 4.1) as a basis for defining coupling patterns (subsection 4.2). Coupling patterns are used within a meta coupling architecture (subsection 3.3.2), and both provide the foundation for various coupling implementations (chapter 5).

[3]The Gang of Four (GoF) are the authors, Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides.

2.2 Coupling in Informatics

In computer science, software consists of a multitude of more or less interrelated software elements. The type of element and its complexity depends on the software concepts[4] and programming paradigms[5] used. In order to solve a given task, complex aspects, such as data schemas and certain behaviors have to be modeled. This can only be achieved with software elements that are able to interact by coupling the interrelated software elements. Coupling in computer science refers to software elements, pieces of software and complete software applications that are able to work together. The coupling can be classified according to the degree of the relationship between the coupled elements (subsection 2.2.1), which has been an area of extensive research in the last two decades (subsection 2.2.2).

2.2.1 Degree of Coupling

Coupling *"is a measure of the relationships among modules [125]"* and *"a qualitative indication of the degree to which a module is connected to other modules and to the outside world [138]."* This is valid for all other coupled software elements as well. The degree of coupling is a measure of the correlations and dependencies between coupled software and software elements. In the literature, these degrees are determined to be either *weak* (also *low* or *loose*) or *strong* (also *high* or *tight*). Two software elements are said to be *strong coupled*, if changes in one software element lead to significant changes in the other, whereas changes in *weak coupled* elements cause fewer or no changes, as shown in Figure 2.3.

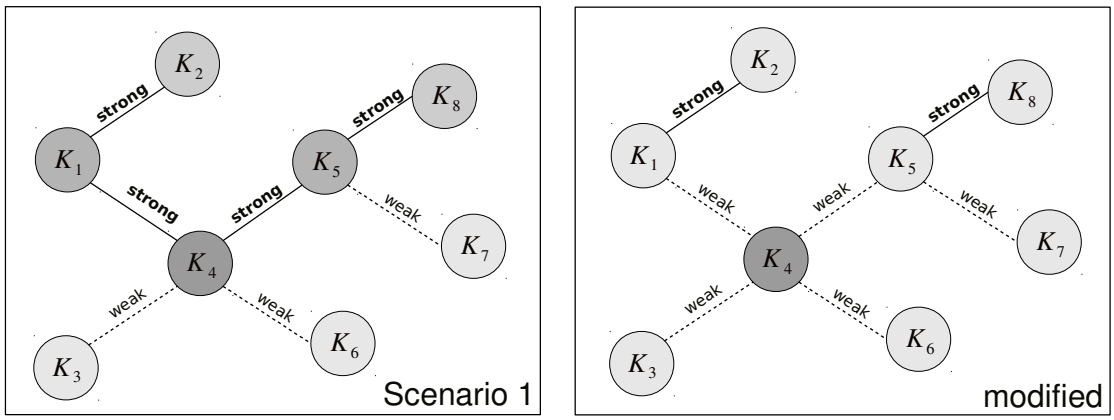


Figure 2.3: Software system consisting of weak and strongly coupled software elements.

[4]E.g., modular programming [132] is based on *modules*; component-based programming [115] on *components*.
[5]E.g., object-oriented paradigm is based on *classes*.

In Figure 2.3, a scenario consisting of eight interrelated software elements is shown.

- With respect to the mostly strong couplings shown in Figure 2.3 (left), changes in software element K_4 directly cause changes in K_1 and K_5, and indirectly in K_2 and K_8. In this scenario, the coupled elements behave sensitively because of the changes in K_4.

- In the modfied version of Figure 2.3 (right), the couplings between K_4 and K_1 and between K_4 and K_5 are replaced by weak couplings. The changes in $K4$ have less influence on the system architecture. In this modified scenario, the coupled elements behave insensitively because of the changes in K_4.

The weaker the degree of coupling is, the more independent the software elements are, which makes their behavior and functional cohesion more insensitive to changes. Software systems with strong coupling are unstable, require more effort to modify, and have decreased reusability. Therefore, weak couplings are an indicator of well-structured software systems and good software design. They are preferred in software engineering in order to develop systems with high readability and maintainability. However, the focus of this thesis is on finding coupling strategies between complete software applications rather than software design. Hence, the degree of coupling is a minor aspect in the context of this thesis.

2.2.2 Classification

The very first classification of couplings between simple software elements was introduced by Myers in 1974 [125]. It consisted of six coupling types in the domain of modular design, as shown in Table 2.2.2. They were then extended, categorized and evaluated by Page-Jones in 1980 [131]. The extension included *tramp coupling*, *bundling* and *hybrid coupling*. This categorization resulted in *normal couplings*,[6] *common couplings*,[7] and *content couplings*.[8] Another classification of couplings was introduced in 1990 by the IEEE[9] [128], which led to IEEE 610[10] [85]. It includes six coupling types, which differ from Myers' classification, however only in some minor aspects. Additional coupling types, e.g., with respect to the object-oriented paradigm, can be found in [138], such as *import coupling*, *routine call coupling*, *inclusion coupling* and *type use coupling*.

[6] Includes data coupling, tramp coupling, stamp coupling, bundling, control coupling and hybrid coupling.
[7] Includes external coupling and common coupling.
[8] Includes content coupling.
[9] The Institut of Electrical and Electronics Engineers is a non-profit professional association.
[10] IEEE Standard Glossary of Software Engineering Terminology.

Coupling Type	Description	Degree
Data Coupling	Data coupling occurs when modules share data and are not content, common, external, control, or stamp coupled.	weak
Stamp Coupling	Stamp coupling occurs if two modules reference the same data structure, providing that this data structure is not global.	weak
Control Coupling	Control coupling occurs if one module passes elements of control as arguments to another module.	weak
External Coupling	External coupling occurs if two modules reference the same externally declared symbol.	medium
Common Coupling	Common coupling occurs when a group of modules reference a shared global data structure.	medium
Content Coupling	Content coupling occurs if one module makes a direct reference to the contents of another module.	strong

Table 2.1: Module couplings by Myers [125] and their degrees by Page-Jones [131].

Unfortunately, the coupling of complex software systems is different from the coupling of simple software elements. The reason for this has to do with the change from a homogeneous software environment to an heterogeneous one. Thus, the proposed couplings cannot be used directly for autonomous software systems. Hence, chapters 3 and 4 focus on the software couplings of complex software systems.

2.2.3 Coupling Quality

Software metrics are an established concept for quality assessment in the domain of computer science. They are a measure of the quality of a software or how its specifications have been designed and implemented. Due to the large number of objective, reproducible and quantifiable measurements, it is difficult to define or measure software qualities and quantities globally; instead, they are limited locally to narrow domains, like budget and schedule planning, cost estimation and software structure. Classical software metrics are used to measure properties, such as the size of a software program (LOC), the time between failures (MTTF), the maximum length of the inheritance tree (DIT) or the number of children (NOC). However, the quality of coupled software systems mainly depends on the quantity and accuracy of the data to be transferred. It is obvious that classical software metrics are not suitable for performing

an assessment of coupled software systems.

Nevertheless, semantic interoperability between software systems has been an area of active research for the past ten years. The SPADEX [22] and PM4D [94] reports highlight the loss of information that occurs between IFC certified software. Similar results are presented by Geiger [68], Bazjanac [25], Dayal [108], and the IAI Forum Denmark [84]. Ma et al. [111] specified a number of changes that occur during the data exchange – entities appeared, disappeared or changed. Further studies published by Pazlar and Turk [134, 135] and the data interoperability benchmark test by Jeong et al. [89] confirmed these results. An excellent review and assessment of interoperability testing methods for product data models used in the construction industry can be found in [106].

However, planning data in civil engineering are sensitive. Lost or incorrect data can lead to problems, therefore, assessing coupling quality (i.e., the quality of the mapping) is important in software coupling. Current processes for evaluating data exchange can be used for assessing coupled systems. Changes resulting from inadequate data mappings are commonly identified a posteriori either by visual inspection or file comparison.

Visual Inspection

In practice, visual inspection is widely applied to give a rough assessment of exchange quality. It refers to the detection of changes in data directly on the screen or on printed drawings by the engineer. It is evident that visual inspection can only detect major problems, like missing or misrepresented elements, as shown in Figure 2.4.

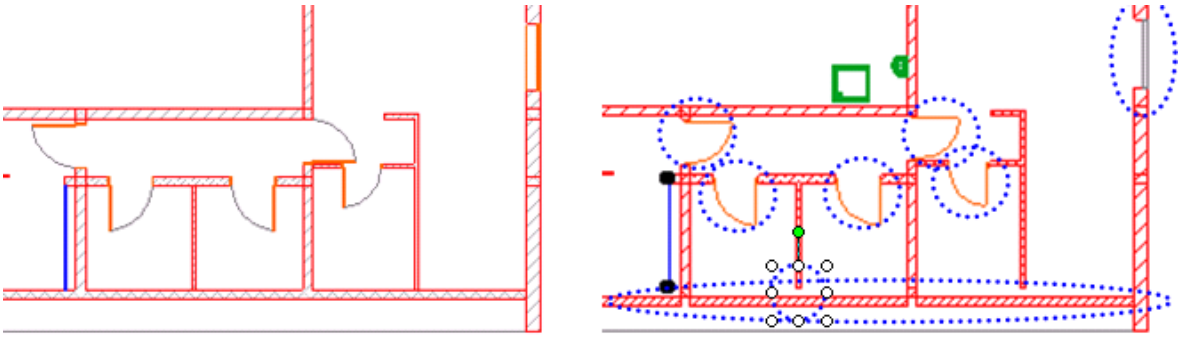


Figure 2.4: Mapping irregularities within a complex model testing, source [135].

File Comparison

File comparison is an approach for assessing the quality of data exchange based on its instances. In contrast to visual inspection, it can detect further minor changes in data. However, this approach is limited to certain conditions. The data exchange must obey a common data format, which should preferably be a standard. In addition, objects have to be unique and distinguishable from one another. Furthermore, the process of mapping and file comparison is time and resource consuming and the results have to be qualitatively evaluated by the user. Hence, it cannot be used for each data transfer and coupling scenario. As an advantage, only the exchange data schema must be known. The assessment process occurs via files, as shown in Figure 2.5. Firstly, the instances of application A to be transferred have to be exported to an external file. Secondly, the file has to be imported into application B. Following these two steps, the objects have to be transferred back to application A. No changes are made to the data during these two steps. In the final step, the changes can be identified by running a file comparison of both external files [111].

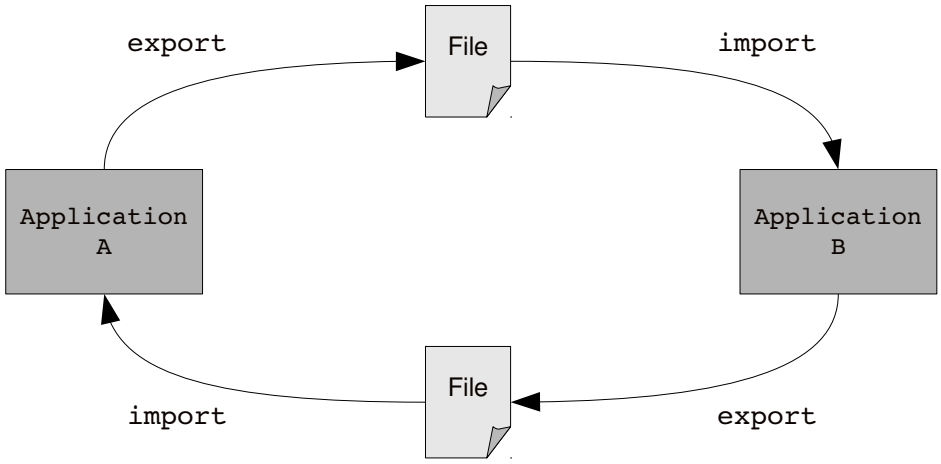


Figure 2.5: File-based information exchange.

In theory, the complete import and export process should be lossless and no modification of data should occur. In practice, however, there is a loss of data, which is inevitable. Amor concluded that with respect to the incompatible data schemas and the many data mappings, perfect semantic interoperability cannot be expected [141]. Gielingh added that with the current generation of product data technology (PDT) standards, loss of data or meaning can hardly be avoided [69]. Other researchers have also highlighted the difficulties involved in enabling semantic interoperability between software systems with different internal schemas

[20, 24, 45, 103, 188]. Vergeest and Horvath went further and made a distinction between the different types of interoperability issues in an information exchange transaction [171]. In the last 30 years, several exchange standards of digital product data have been developed (see subsection 2.3.1). Especially in the construction sector, the use of the *IFC*[11] has led to a new trend called Building Information Modeling (BIM). A special software system called *EVASYS*[12] has been developed in order to provide a quick and easy way to compare two IFC files automatically [111]. In general, changes are listed according to certain criteria, such as differences in physical file size and number of instances, as well as inconsistent object types and attribute values, and schema inconsistencies. Another interpretation in the form of software metrics can be found in [101]. The result of a file comparison with respect to a simple IFC wall entity is shown in Figure 2.6.

Application	ADT(N)	ADT*	ARC	ALL	ALL(N)	ALL*	ARC	ADT	ARC(N)	ARC*	ADT	ALL
Native file size (bytes)	118980				49174				508688			
IFC file size (bytes)	3756	5080	5044	3198	3253	3253	5598	5218	5631	5635	7113	3577
Difference in file size (%)	-	35.3	34.3	-14.9	-	0	72.1	60.4	-	0.0	26.3	-36.5
Entities – total**	69	98	84	53	53	53	84	98	84	84	127	60
Entities - diverse***	35	37	36	35	35	35	36	37	36	36	38	36
Entities with GUID****	8	12	12	11	8	8	12	12	12	12	18	10
IfcDirection((0,0,1))	5	3	1	3	3	3	1	3	1	1	4	4
IfcShapeRepresentation	Bound-ingBox	Bound-ingBox	Bound-ingBox				Bound-ingBox	Bound-ingBox	Bound-ingBox	Bound-ingBox	Bound-ingBox	
	Curve2D	Curve2D	Curve2D	Curve2D	Curve2D	Curve2D	Curve2D	Curve2D	Curve2D	Curve2D	Curve2D	Curve2D
	Solid	Brep - Face	Solid	Solid	Solid	Solid	Solid	Brep - Face	Solid	Solid	Brep – Face	Solid
Entities required*****	13	33	13	10	10	10	13	33	13	13	33	10

Figure 2.6: File size and entity comparison [135].

However, differences in the physical file size or the number of instances are not a strong indicator of coupling quality. Inconsistent object types and attribute values are only important if they are needed in the software application to be coupled, and if a loss were to influence the coupling results. Each change listed has to be qualitatively evaluated by the user. Thus, an assessment approach resulting in a global quality value is recommended. It can then be used directly by the user to assess the data exchange. For this reason, a generic a priori approach for assessing coupling quality is introduced in chapter 6. In contrast to the file comparison approach, it operates directly with the involved schemas, is not limited to a common exchange format, and takes the various mapping patterns from the literature into account. An essential issue with this approach is the process of schema mapping.

[11] Industry Foundation Classes: exchange data schema for three-dimensional building and construction data.
[12] EXPRESS Evaluation System.

Schema Mapping

Schema mapping (Figure 6.4) is widely used in applications that involve data sharing or data transformation and plays a central role in data exchange and integration [100]. In general, a schema contains data structures and their relationships [62]. Schema mapping typically describes the relationship between two schemas and can be understood as a triple consisting of a source schema, a target schema and a set of relationships between the source and the target schema.

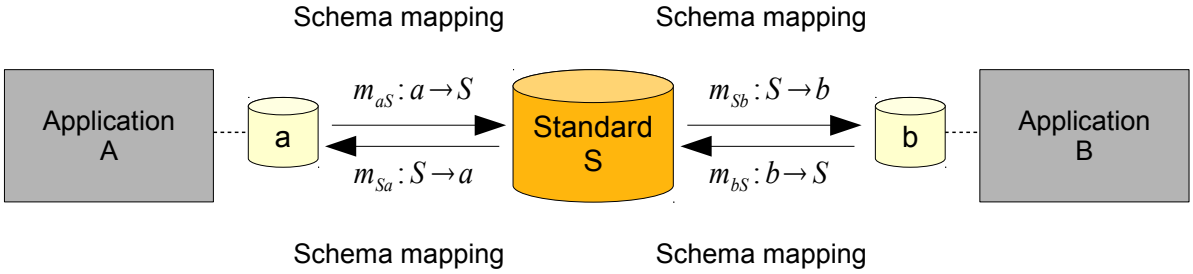


Figure 2.7: Schema mapping approach.

An important aspect of schema mapping is the detection of schema overlaps, which is also known as schema matching. This can be done directly or indirectly by using a standard schema. It should be noted that the quantity of data to be exchanged between two software applications cannot be increased by a standard. A *"data exchange using standardized neutral models appears not to be possible without errors and information losses [69]."* Figure 2.8 shows the resulting set of exchangeable data for different schema configurations, where A and B are the data schemas of the software applications to be coupled and S is the data schema, which is defined by a standard. An additional loss of information may occur (Figure 2.8 - middle) if $(A \cap B) \setminus S \neq \emptyset$. If $(A \cap B) \setminus S = \emptyset$ (Figure 2.8 – right), the exchangeable information is the same as shown in Figure 2.8 (left).

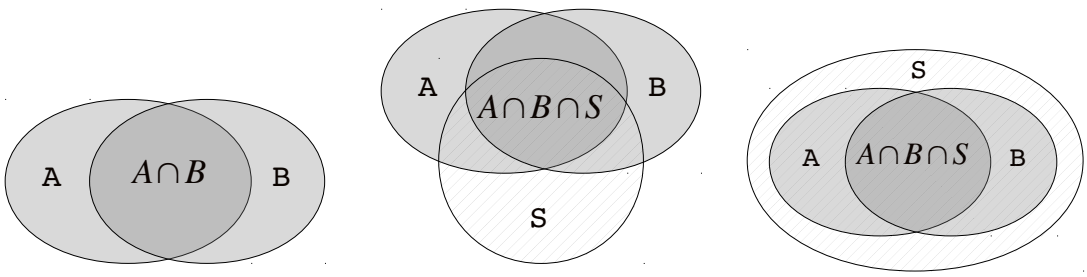


Figure 2.8: Schema overlaps: direct (left), indirect (middle) common, and (right) best case.

Schema matching is investigated extensively in various database application domains. A survey of approaches for automatic schema matching is given by Rahm and Bernstein [146]. A comparison of schema matching evaluations is introduced by Do et al. [43]. The schema mapping can be described by well-known mapping patterns [9, 30, 77, 136]. They describe how data structures of the source schema are related to corresponding data structures of the target schema. Approaches to mapping languages in engineering domains have been developed by [46, 98, 172]. Schema mapping and mapping problems are examined in detail, especially in the field of database management and design [51, 66, 112]. Overviews of the advances made in this field are given by Kolatis [100] and Lenzerini [102]. However, Bakis [23] noted that the capabilities of mapping languages are limited since they only support the structural translation of semantically identical information. He further noted that in order to reconcile any semantic differences, a computer interpretable description of the semantics, as well as a logic for converting between semantics is required and has to be explicitly specified. The implementation of a semantically correct mapping of the overlapping schemas used in architecture, engineering and construction – such as IFC and STEP – is quite challenging. Due to the complexity of engineering tasks, the schemas involved have hundreds of data structures with thousands of attributes and relationships. Amor [10] found that without some definition of a mapping, it is impossible to guarantee the correctness of any translator implemented; he also illustrated and discussed the development of a suite of mapping support tools to ensure semantically correct mappings. Furthermore, he recommends developing a range of certified mappings. A semantic mapping between CAD and IFC property definitions was investigated in [183]. However, one of the major problems regarding the interoperability of systems in the construction industry is the difficulty in assessing accurate data, information, and knowledge in a timely manner at every phase of the construction project life cycle [158].

Evaluated schema mappings would reduce most of the disadvantages of the file comparison approach; and would allow an a priori assessment of semantic interoperability in a timely manner. Therefore, chapter 6 introduces an approach for performing evaluated schema mapping. It is mathematically formalized by using the set theory and graph theory and by taking the various mapping patterns into account. Uncertainties arising within the mapping process are considered as well. Chapter 7 includes the basic principles of performing a schema analysis. They are essential for schema mapping due to the fact that the data structures of a data schema can be interrelated to describe more complex facts. These principles take certain types of relationships with respect to the object-oriented programming paradigm into account.

2.3 Interoperability

Interoperability can be understood as the ability of systems, products or organizations to work together. In the domain of computer science, interoperability is strongly related to systems integration and collaboration with regard to communication and information sharing, and therefore, for achieving a common objective. Due to the complexity of tasks in the construction industry, the multiple phases of a project's life cycle, the many multidisciplinary teams and heterogeneous software environments involved, systems integration is an important prerequisite for achieving collaboration [158] and enabling software coupling. The importance and need for interoperability in the construction industry is shown in various surveys [48, 64, 149]. A study published by the NIST[13] [123] confirms this need and identified a loss of around $16 billion in 2002 resulting from the inadequate interoperability of software systems in the US capital facilities industry. An excellent literature review on systems integration and collaboration in the domain of construction industry is given by Aouad et al. [16] and Boddy et al. [33]. A more critical view on the state of the art in systems integration and collaboration technologies is presented by Shen et al. [158]. Special reviews on standard building data models and model mapping languages [23], storage and exchange mechanisms for building information models [87], process integration of distributed construction processes [127], or the advancement/development of tools to improve BIM technologies [39] are also available. An international perspective of the problems and requirements in the areas of systems integration and collaboration technologies in the construction industry are presented by FIATECH [55] (North American perspective), by the ROADCON project [149] (European perspective), and by Moum et al. [122] (Denmark). In [158], systems interoperability is divided into two different types: *data interoperability* and *frameworks interoperability*. Another classification [23], breaks systems interoperability down into three types, as follows: *conceptual level*, *physical level* and *data management*. The conceptual level, which is similar to data interoperability, is concerned with standard product data models. The physical level and the data management classification can be assigned to frameworks interoperability. They are concerned with the technologies and mechanisms used to enable the physical exchange of data and its consistency. A comprehensive literature review in the field of CIC[14] has been carried out in [33]. The results are categorized at the *data application level*, *application semantic level*, *data process level*, and *process semantic level*. In the context of the thesis, interoperability has been categorized with respect to [158] into data and frameworks interoperability.

[13] US National Institute of Standards and Technology.
[14] Computer Integrated Construction.

2.3.1 Data Interoperability

"Data interoperability is the ability that data generated by any one party can be properly interpreted by all other parties. It is the first step towards any systems integration and collaboration. The enabling technology for data interoperability is data modeling [158]." Due to the various proprietary data models developed by vendors, organizations or consortia, data interoperability plays an important role for software coupling in the construction industry. A data model describes the data of a certain domain through the definition of application objects, constraints and relationships between objects. *"A common neutral model is the most feasible solution in AEC/FM to enable data sharing or integration in heterogeneous applications [158]."* Gielingh stated that *"organizations can benefit from the exchange or sharing of digital product data across the borders of disciplines, organizations and vendor-specific solutions [69]."* Teeuw et al. further concluded that *"it is worthwhile to use international, open standards for data communication [163]."* For the last 30 years, various data models have been developed by international organizations[15] or industrial consortia.[16] A review on data interoperability through product sharing is given in [23]. Due to the wide range of different tasks and domains in the construction industry, many different neutral data models and standards have been developed for the exchange of two-dimensional CAD data and three-dimensional building models, to describe geographic information and services, or to exchange data in the structural steel and precast concrete industry. Most of them are described in ISO 10303 (STEP). Most of the data models today follow the object-oriented approach in order to describe common information in a hierarchical inheritance tree.

DXF – Drawing Interchange Format

DXF[17] was originally developed as a proprietary data exchange format between AutoCAD and other CAD software systems. It supports common geometrical objects (Point, Line, Arc, Circle, Spline, etc.), text elements, external fonts, symbols, dimensioning, layers, freeform surfaces and solids. The first version was initialized in 1982. The first publication of DXF specifications started with AutoCAD Release 13. DXF became a de facto standard primarily for the exchange of two-dimensional CAD data. Other CAD data exchange formats are the Industry Foundation Classes (IFC), the Standard for the Exchange of Product Model Data (STEP) and the Initial Graphics Exchange Specification (IGES).

[15] E.g., ISO – International Organization for Standardization.
[16] E.g., IAI – International Alliance for Interoperability.
[17] Developed by Autodesk, also known as Drawing Exchange Format.

IGES – Initial Graphics Exchange Specification (NBSIR 80-1978)

IGES[18] is a neutral product data format for an exchange of data between CAD systems, e.g., traditional two-dimensional drawings, three-dimensional simulation and analysis models and for manufacturing. It was used extensively in the automotive, aerospace, and shipbuilding industries. However, with the first release of STEP (ISO 10303) in 1994, interest in deloping the IGES decreased; and the last version was published in 1996. STEP is currently taking over the role of IGES and remains the most widely used standard for CAx and PMI interoperability [158].

IFC – The Industry Foundation Classes (ISO 16739)

The Industry Foundation Classes (IFC) are a neutral, object-oriented set of product data model specifications for the exchange of three-dimensional building and construction data in the AEC/FM[19] industry. It was developed by buildingSMART International [34], which was formerly the International Alliance for Interoperability (IAI). It provides the foundation for a new trend in construction industry: the Building Information Modeling (BIM). Today, the IFC are a feature in the most ot the CAD software applications on the market [17, 18, 19, 170]. It is applied in domains, like construction scheduling and change management [165], integration of extensible life cycle construction product data [113], architectural/structural design and construction [92, 109, 110, 137, 181], cost estimating [63], HVAC engieneering, simulation [144], and facilities management [154, 169, 184]. The IFC has become widely recognized as the common data exchange format for interoperability within the AEC industry [47].

CIS/2 – CIMSteel Integration Standards

The CIMSteel[20] Integration Standard [41] is an industrial standard and product data model for the electronic data exchange of structural steel project information. It is intended to enable a seamless information flow between all members involved in the construction of steel frame structures and to support the analysis, design, and detailing of steel frames. It is defined in EXPRESS and fully harmonized with STEP, like the IFC. In [95], the extent to which CIS/2 can specify product descriptions capable of supporting the automated erection of structural steelwork was investigated.

[18] Digital Representation for Communication of Product Definition Data was first published in 1980.
[19] Architecture, Engineering, Construction/Facilities Managment.
[20] Computer Integrated Manufacturing of Constructional Steelwork, which is also denoted as the Logical Product Model.

Integration of life cycle data for process plants (ISO 15926)

ISO 15926[21] is a standardized generic data model for sharing and exchanging data throughout the entire life cycle of a facility. It was originally developed for the oil and gas industries to achieve data integration and interoperability. It has been in development since approximately 1992 (initially called STEP AP221), consists of seven parts and is near completion. ISO 15926 enables the sharing and exchange of functional requirements, physical solutions, types of objects and individual objects, as well as activities throughout all life cycle stages, supply chain company types and all types of facilities, such as industrial, commercial, institutional and residential [4]. Like IFC and CIS/2, the data model of ISO 15926 is based on EXPRESS.

Problem Statement

The problem of sharing necessary information between software applications is depicted by the information processing cycle shown in Figure 2.9. The information to be exchanged has to be abstracted (decoded) into the various proprietary and optimized exchange data models. During this abstraction process, a loss of information can occur. In the next step, the information decoded in the exchange data model then has to be interpreted (encoded) again through the software application to be coupled. In this interpretation process, a further loss of information can occur. Furthermore, the data models of the software applications to be coupled are mostly not equal and thus, have to be mapped to each other. This can result in a further loss of information.

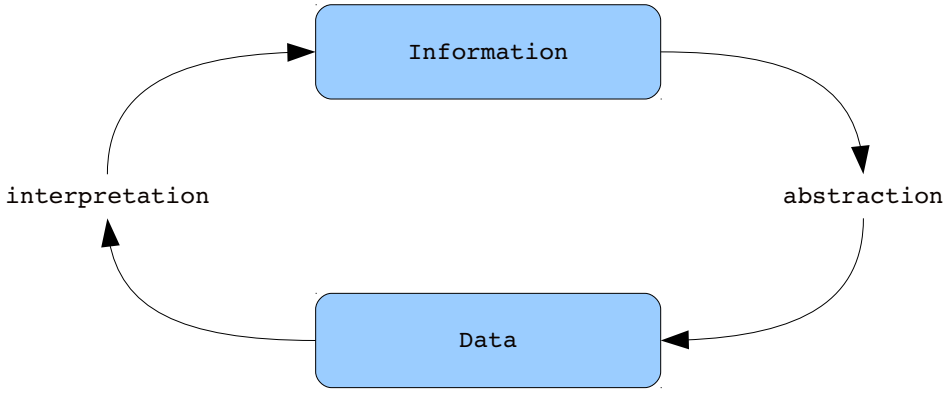


Figure 2.9: Information processing on the computer level.

[21] Industrial automation systems and integration – Integration of life cycle data for process plants, including oil and gas production facilities.

Unfortunately, due to the mostly proprietary and incompatible data schemas, several data mappings and data conversions between the data structures used are required. The research of Banerjee et al. [24], Lerner and Habermann [103], Eastman [45], Zicari [188] and Atkinson et al. [20] highlight the difficulties of semantic data interoperability between applications with different internal schemas. *"The major problem facing today's data interoperation solutions is the existence of different exchange flavors. A flavor of a standard is evidenced when two different vendors interpret the same standard in two different ways. Since BIMs are highly complex, it is unavoidable [158]"* and *"data exchange using standardized neutral models appears not to be possible without errors and information losses [69]."* Various reports [22, 94] and publications [89, 105, 111, 135, 141] confirm these statements. To do this, the error-free exchange of data is a prerequisite and one of the main challenges in achieving collaboration and distributing work across disciplines and organizations. Data interoperability plays an important role in data coupling, which is one of the most used coupling concepts in civil engineering. In addition, data coupling is a root element of the coupling graph (section 3.2) and thus the foundation for various data-based coupling strategies. Furthermore, data interoperability is an important factor in assessing coupling quality. Therefore, a generic a priori strategy for the qualitative assessment of data interoperability and data exchange is introduced in chapter 6 and its adaptation to the object-oriented paradigm is shown in chapter 7.

2.3.2 Frameworks Interoperability

Frameworks interoperability refers to the technical and technological aspects to consider for achieving systems integration. This type of interoperability is required when software systems have to communicate and collaborate. For this reason, internet-based communication technologies and protocols have been developed. Frameworks interoperability has assumed a key role in the AEC/FM industry, since *"collaborative working in construction is becoming widely spread as many activities are performed globally with actors based in various geographical locations [52]"* and *"the need to develop and deploy comprehensive software systems that can support this integration has become evident [110]."* The capabilities of such software systems range from simple data sharing/exchange/management to full integration of data and processes across all stages of a construction project's life cycle. Research projects, such as ATLAS [143], VEGA [167, 186], RATAS [26, 27], COMBI [145], OPIS [65], ToCEE [142], and the ones listed in Table 2.3.2 have been carried out in order to investigate the methods, technologies and infrastructures required to develop such comprehensive environments. A more detailed overview of these research projects can be found in [16, 33, 136, 140, 166].

Extensive research on the integration of the different project life cycle phases, such as project design, cost estimating, virtual prototyping, construction planning, simulation have already been done [14, 40, 65, 88, 91, 168, 182].

Project	Domain	Aim/Result	Literature
WISPER	design, visualization, estimating, planning, specifications, supplier information	development of a Web and IFC-based collaborative working environment	[52, 53]
ICON	design, procurement, construction management	to define contextual and conceptual models for an integrated database	[13, 14]
OSCON	architectural design, cost estimating, process management	information modelling, sharing, integration via central database	[13, 15]
COMMIT	construction	to improve long-term effectiveness using an information management model	[7]
COMBINE	simulation (thermal, energy, comfort), HVAC, cost estimation, design	development of future intelligently integrated building design systems	[70, 139]

Table 2.2: Overview of research projects.

With respect to the rapid development of communication technologies in the last two decades, various systems integration approaches and concepts have been developed and applied within the different engineering domains. Shen et al. [158] divided systems integration into the follownig classifications: *web-based*, *distributed object/component-based*, *agent-based*, and *web service-based systems*.

Web-based systems

Web-based systems follow a centralized approach to enable distributed teams to share information via the World Wide Web[22] in a quick and easy way all over the world. The WWW started in 1991 and was originally developed to share information, knowledge and ideas through interlinked hypertext documents via the internet. Sharing and accessing hypertext

[22] Abbreviated as WWW or W3.

documents is achieved through client-server architectures that communicate over a Hyper-Text Transfer Protocol (HTTP). The hypertext documents can then be viewed using special web browsers. Web-based systems can be implemented in a timely manner through simple client-server architectures. They are commonly used to facilitate accessing information over the complete life cycle of the planning process. Several industrial surveys [48, 64, 149] show that most of the IT tools used for solving engineering tasks integrate web-based systems, like basic technology.

- *In the field of coastal engineering, web-based systems are used within the NOKIS++ project [59] in order to provide digital atlases [79] for bathymetry, tide, wind and swell prognoses.*

- *In the field of construction industry, web technology is used for the following: to implement web-enabled collaborative building design [40], to simplify the manipulation of files [168], to provide web-based services for information management and exchange [180, 182], for fire simulation [88], or to support distributed designing, visualization, estimating, planning, specifications and supplier information [53].*

- *In the domains of aeronautics, mechanics and architecture, web-based database servers [32, 40, 50, 52, 53, 96, 137, 162] are used to manage the huge amount of data (e.g., building data) and to avoid inconsistency, poor accessibility, integrity, and the authority arising from conventional file-based systems [5, 160].*

- *Web technology in the field of computer integrated construction has been applied in certain research projects [13, 53, 167, 186, 187], mainly in the area of data sharing and exchange. A detailed view of a collaborative building modelling project using model servers based on the IFC can be found in [90].*

However, the planning process is characterized not merely by information sharing and management, but also by the interaction between the engineers. With respect to the complexity of engineering tasks, the multiple phases of a project life cycle, and the many multidisciplinary teams involved, the interaction between engineers is essential for coordinating and solving complex engineering tasks, such as supporting collaborative designing and modeling, providing analysis and simulation services, or managing projects. These requirements are hardly implementable with basic web technology. Therefore, simple web-based systems are more suitable for daily construction project management and document sharing rather than for complex engineering problems [158].

Distributed object-based systems

Distributed object-based systems extend object-oriented programming systems to distribute objects across a heterogeneous network. Although the objects are distributed on different computers throughout a network and exist within their own address space outside of an application, they interact as a unit and appear as local objects for an application. The three most popular approaches, i.e., *CORBA*[23] [75], *DCOM*[24] [118], *RMI*[25] [129], are primarily used to achieve systems integration on a higher level. The web is being transformed from a *"static, two-tier, client-server, unidirectional environment for the publishing and broadcasting of electronic documents into a full-blown client-server medium with the potential to run line-of-business applications and to deal with the complex requirements of multistep business-to-business and consumer-to-business transactions [53]."*

- CORBA is based on the Internet Inter-ORB Protocol (IIOP) and allows separate pieces of software to work together and behave like a single software application/system. The separate pieces of software can be written in different programming languages, which run on distributed computers and/or on different operating platforms. However, before an object can be remotely shared and distributed, the interface that describes the object has to first be translated in the Interface Definition Language (IDL), a part of the CORBA specification. Then the object's IDL interface has to be mapped and compiled to the implementation language of the software application used. This is available for the majority of programming languages. Finally, the CORBA objects can interact via the Object Request Broker (ORB). It acts as an object bus and enables communication and interaction between the registered CORBA objects, including finding, as well as receiving requests and replies.
  - *Within the WISPER project [53], CORBA has been used to enable transparent data sharing and exchange through an integrated computer environment. It supports engineers in the area of design, visualization, planning, interaction and distributed access to applications [52].*
  - *Within the VEGA project [186, 187], a CORBA-based middleware has been used to enable remote access on distributed STEP-based geometrical product instances. It enables 3D viewing of distributed building objects previously designed under a CAD tool [185].*

[23] Common Object Request Broker Architecture developed by the Object Management Group (OMG).
[24] Distributed Component Object Model developed by Microsoft.
[25] Remote Method Invocation developed by ORACLE, formerly JavaSoft.

  - *Within the OSCON project [13], interactive system OSCONCAD has been developed for integrating CAD and construction-related applications. CORBA technology has been used for the distribution of objects among construction applications for function and data sharing [54].*
  - *Within the COMMIT project [7], a set of CORBA compliant distributed components are used in order to provide the intelligent integration of information throughout all stages of a construction project's life cycle.*
  - *Within the SFB 524 [114], a descriptive integration platform has been developed to support cooperative, process-oriented aspects of the architectural, static and constructive conversion of buildings. CORBA has been used to define a set of fundamental data types and methods [130] for modelling and exchanging building inventory independently of specific programming languages and software applications.*

- DCOM is a proprietary Microsoft technology and enables communication among distributed COM-based objects. However, compared to CORBA, COM interface technology has two major disadvantages. It has been defined and implemented as a standard only on Microsoft's Windows platform and it is limited to Microsoft's .NET programming languages, such as C#, Visual Basic, or C++/CLI.
  - *Within a middleware solution called BSPro ComServer [12], DCOM technology has been used to enable easy access to building geometry within IFC files and to link new and/or existing software tools in the building services domain.*
  - *Within a three-tier component-based framework [110] for facilitating the implementation of modular and distributed integrated project systems, a set of middle-tier components have been implemented using DCOM technology in order to support building design and construction projects.*

- RMI is a specific Java Application Programming Interface (API). It allows remote method calls of Java objects that are distributed on different computers and living in different runtime environments, called the Java Virtual Machine (JVM). The first step is to create a remote interface indicating that a Java object is a remote object whose methods can be invoked across virtual machines. Only those methods defined in the remote interface are available to be remotely called. As a feature, RMI not only allows remote method calls of Java objects, but also remote method calls of CORBA objects in order to achieve interoperability between RMI and CORBA applications.

Agent-based systems

"An agent is a computer program capable of flexible and autonomous action in a dynamic environment, usually an environment containing other agents [107]" and *"software agents are critically needed to meet the dynamic changes in information technology, which inhabits vast amounts of discrete information that require complex processing under uncertain conditions in order to abstract knowledge and make decisionsv[49]."* In contrast to classical objects, a software agent *"possesses special intelligent properties, such as autonomy, reasoning, mobility, sociability, learning ability, cooperation, and negotiation. This allow agents to initiate their actions without the need of direct intervention or guidance of humans or other entities, and to interactively cooperate and communicate with each other, and with their environment to accomplish special tasks that cannot be performed by conventional software [49]."* This means that they do not need to be invoked externally because they act autonomously and take action as they deem appropriate. Software agents are able to perceive changes in their internal and external environment in order to react and compensate for those changes in an appropriate manner. Hence, they are *"best suited for applications that are modular, decentralized, changeable, ill-structured, and complex [133]."* A detailed overview of agent technology and a platform for researchers and developers with a common interest in agent technology can be found in [6]. Software agents can be classified according to their properties, as well as according to their tasks and roles. Nwana [126] mentions several dimensions in the classification of existing software agents and identifies seven types: *collaborative agents*, *interface agents*, *mobile agents*, *information/internet agents*, *reactive agents*, *hybrid agents* and *smart agents*. Another classification of software agents, which classifies them as either *weak* or *strong* is suggested by Woodridge and Jennings [179]. Agents can be incorporated in Multi-Agent Systems (MAS), in which they are capable of mutual interaction to solve more complex problems that are beyond their individual capacities or knowledge.

Software agents are used in various sub-disciplines of information technology[26] [107] and for solving industrial problems, such as the coordination, simulation, scheduling and controlling of designers, processes or products [133]. In civil engineering, the agent-technology is used to integrate data, information, and knowledge captured and accumulated during the entire facility life cycle [159], to improve collaboration within the structural design [31, 153], to support network-based fire engineering [164], to monitor security-relevant structures [120], or to compute and optimize finite elements [124].

[26] E.g., computer networks, software engineering, artificial intelligence, human-computer interaction, etc.

Web services/Semantic Web

A web service is *"a software system designed to support interoperable machine-to-machine interaction over a network [71]."* In contrast to basic web servers, which are more or less passive, web service technology enables active/proactive communication and sharing of data/information in order to build cooperative, coordinated software systems. Hence, *"Web services are emerging as a major technology for deploying automated interactions between distributed and heterogeneous applications [29]."* In addition, multiple web services can be combined via the Web API [176] into so-called mashups. A *"mashup simply indicates a way to create new Web applications by combining existing Web resources utilizing data and Web APIs. Mashups are about information sharing and aggregation to support content publishing for a new generation of Web applications [29]."* Another web technology that is used for systems integration and collaboration is the Semantic Web. The Semantic Web is basically an extension of the current World Wide Web, which, up to now, has concentrated on the interchange of documents, and acting as a platform for the integration and combination of data drawn taken from diverse sources. *"Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data [177]."* The *"Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is based on the Resource Description Framework (RDF) [178]."*

Conclusion

There are many data models and communication technologies available for accomplishing the interoperability required for the coupling of heterogeneous software systems. However, the description of coupling strategies using design patterns only exist for partial solutions and technical aspects that are limited to the overall coupling process. As a consequence, a coupling graph and coupling pattern language consisting of templates for defining coupling patterns are introduced in chapters 3 and 4.

3 Modeling of Coupling

From a computer science point of view, a software application consists of a multitude of more or less complex software elements, such as classes, modules and components. In order to solve complex tasks and describe the required data, the software elements have to interact. In order to implement these interactions, the interrelated software elements have to be coupled. Coupling within homogeneous environments has been extensively examined in recent years and can be accomplished with different coupling types [61], as briefly introduced in subsection 2.2.2. However, from an engineering point of view, the planning process of buildings is highly complex and therefore broken down into smaller, manageable tasks. These tasks then can be solved separately and concurrently by different engineers through the use of specialized software applications. Inevitably, this leads to a great deal of communication between the engineers and consequently between the software applications used. Unfortunately, the coupling of software applications is different from the coupling of software elements within a software application. The reason for this is the change from a homogeneous to a heterogeneous software environment. In general, software applications use their own data structures, which are optimized to solve the task. In addition, they might run on different operating systems and computers, or implemented by different programming languages. These languages might also be based on different programming paradigms. Furthermore, softwares are influenced by the rapid developments in computer science. In general, new technologies and paradigms are developed in short cycles. They are used and adopted by the software to improve and to extend its functionality.

In this chapter, the basic principles for defining a coupling pattern language are introduced. They enable the formulation of coupling strategies for software coupling by taking different coupling aspects into account (section 3.1). However, coupling aspects cannot be considered individually because they must be combined. Thus, they are integrated in a coupling graph (section 3.2). Finally, the multiplicity of coupling aspects and technical software methods that exist have resulted in a great variety of coupling strategies. Therefore, additional abstraction mechanisms are needed, therefore, a metamodel archeterogeneoushitecture (subsection 3.3.2) has been used.

3.1 Coupling Aspects

Coupling aspects are important in software development. They have to be taken into account when developing new coupling strategies or selecting adequate coupling strategies from existing ones. In order to do this, three fundamental questions need to be considered:

- ***How can software applications be coupled?*** The answer depends on technical aspects, such as the flow of data (subsection 3.1.1), the degree of communication (subsection 3.1.2), the synchronization of resources (subsection 3.1.3), or the runtime behavior of software systems (subsection 3.1.4). The technical challenge here are the different programming languages, operating systems and distributed software applications that have to be considered. Technically, software coupling is well supported by a variety of software technologies (subsection 2.3.2). Technical software coupling is necessary for further semantic software couplings.

- ***What information has to be coupled?*** The answer to this question depends on the semantic aspects, such as the exchange of data and functionality (subsection 3.1.5). The semantic challenge here are the various data schemas (subsection 2.3.1) and the integration of external functionality that have to be taken into account. The correct mapping of data is a particularly important factor (subsection 2.2.3). Semantically, software coupling is poorly supported and thus the coupling has to be reimplemented for each coupling scenario. Unfortunately, successful semantical coupling is necessary for useful software couplings.

- ***What is the quality of the coupling?*** The answer to this question depends on the qualitative aspects, which have been extensively examined for data coupling (chapter 6). It can be used by the engineer as an indicator of the level of confidence of the results that are generated by coupled software applications.

  - From a technical point of view, the quality of software couplings is 1 if the software applications are connectable, and 0 if they are not connectable. A technical coupling quality of 1 is the prerequisite for further semantic couplings.

  - From a semantic perspective, the quality of software coupling depends on the semantic overlaps of data schemas (Figure 2.8 on page 18), and on mapping them successfully (subsections 6.3.2 and 6.3.3). The quality of semantic coupling can be described by a variety of measures, such as single values (subsection 6.8), ranges of values (subsection 6.16), or linguistic terms.

3.1.1 Data Flow

In this subsection, the technical aspect of data flow between software applications is considered, which can be defined by two aspects. The first aspect is the function of the communication partners, which can be differentiated by whether data is delivered or retrieved. The second aspect is the direction of the data exchange, which can be differentiated by whether the data flow is unidirectional or bidirectional. In the context of this thesis, coupling based on data flow is classified into *unidirectional* and *bidirectional coupling*.

Unidirectional Coupling

Unidirectionally coupled software applications are only allowed to exchange data in one direction without the need for feedback. Two software applications are considered to be unidirectionally coupled if the data flow only occurs in one direction. The function of the communication partners as shown in Figure 3.1 can be the delivery of data (A delivers data to B) or the retrieval of data (C retrieves data from D). Unidirectionally coupled software applications cannot be used to describe iterative and interactive events.

Figure 3.1: Unidirectional data flow between two software applications.

Bidirectional Coupling

Bidirectionally coupled software applications exchange data in both directions. Two software applications are considered to be bidirectionally coupled if the data flow occurs in both directions. The function of the communication partners as shown in Figure 3.2 can be the delivery of data, such as when A delivers data to B (e.g., geometry data) and waits for a response from B (e.g., simulation results), or the retrieval of data, such as when C retrieves data from D (e.g., geometry data) and sends data as a response to D (e.g., simulation results). In contrast to unidirectional coupling, bidirectionally coupled software applications can be used to describe iterative and interactive events.

Figure 3.2: Bidirectional data flow between two software applications.

3.1.2 Degree of Communication

The degree of communication describes the communication and data exchange between a group of software applications. It is defined by the communication types found in computer networking, as shown in Figure 3.3. It enables the implementation of highly distributed software environments, e.g., client-server architectures. In the context of this thesis, the communication and data exchange within such distributed environments is denoted as *multidirectional coupling*, which can be classified into the following relations: *unicast*, *anycast*, *multicast*, and *broadcast*. Unicast and anycast are *one-to-one* relations; and they are used for the single addressing of software applications. Multicast and broadcast instead are *one-to-many* relations; and they are used for the multi addressing. However, multidirectional coupled software environments can also be broken down into sets of unidirectional and bidirectional couplings.

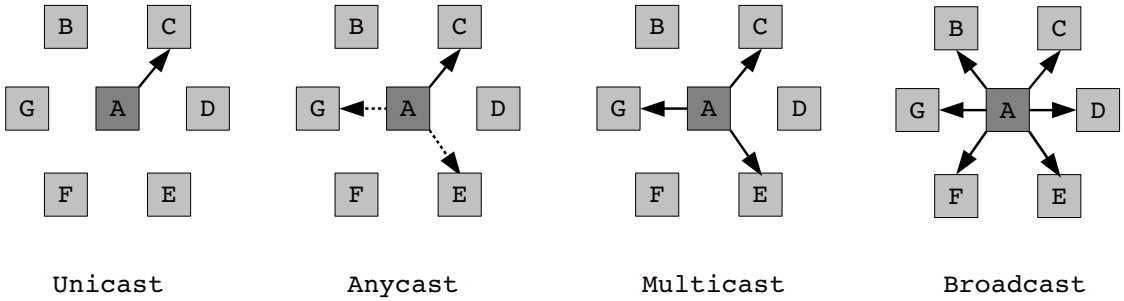


Figure 3.3: Communication types between a group of software applications.

- Unicast communication occurs when the server always addresses the same client from a group of clients. If the client addressed is not available, the server has to wait until the client is available.

- Anycast communication occurs when the server addresses exactly one client from a group. The choice of which client is addressed takes place automatically through a condition. If the client addressed does not fulfill the condition, the server addresses another client that fulfills the condition.

- Multicast communication occurs when the server addresses only a subset of clients. The communication to the selected clients takes place simultaneously in a single transmission.

- Broadcast communication occurs when the server addresses all the clients in the group. Broadcast is a special type of multicast.

3.1.3 Synchronization

Synchronization is another technical aspect. It is closely associated with terms like coherence, consistency and integrity. Synchronization can occur with respect to data, communication and processes. The synchronization of data refers to establishing coherence among datasets in order to achieve data integrity. It is an important issue in database management systems,[1] which is supported by special database transactions. The synchronization of communication is when all the communication partners involved are present at the same time in order to achieve direct communication. It can be supported by synchronous messaging used within company board meetings, chat room events and instant messaging. The synchronization of processes refers to enabling temporal harmonized interactions of interrelated processes in order to achieve process coherence. Process synchronization is used to implement data and communication synchronization. A coupling according to synchronization can be classified into the following two types: *synchronous* and *asynchronous coupling*.

Synchronous Coupling

Synchronously coupled software applications are temporal in relation to the data, communication or processes. Two software applications are said to be synchronously coupled if the resources or processes that have to be shared are synchronized. Synchronous coupling can be implemented to avoid the simultaneous use of common resources, such as global variables or databases. The term mutex or mutual exclusion describes the different synchronization mechanisms used in the domain of concurrent programming, such as lock, semaphore or monitor.

Asynchronous Coupling

Asynchronous coupled software applications are out of sync with regard to data, communication or processes. Two software applications are said to be asynchronously coupled if the resources or processes that have to be shared are not synchronized. Access to a resource occurs without mutex. Asynchronism with respect to the data between coupled software applications presents a danger because the datasets to be shared can become inconsistent. Asynchronism with respect to the communication between coupled software applications results in time-staggered and unblocked communication. Asynchronism with respect to the interrelated processes between coupled software applications presents a danger because they can become temporally unharmonized.

[1] Available for relational data models (RDBMS – Relational Database Management System) or for object models (OODBMS – Object-Oriented Database Management System).

3.1.4 Runtime Behavior

Runtime behavior is another technical aspect. It is defined by the life cycle of softwares. Runtime refers to the time during which a software application is running. Coupling according to runtime can be classified into the following two types: *online* and *offline coupling*.

Online Coupling

Two software applications are said to be online coupled if the coupling strategy used requires both applications to be running. An implementation needs to fulfill special technical preconditions, such as having an connection (communication channel) between both. This can be achieved with standardized communication protocols. In addition, one software application has to act as a listener/receiver and the others as a sender/transmitter. Finally, online coupling can be implemented in different ways, as shown in Figure 3.4), e.g., centralized (left), decentralized (middle), and hybrid (right).

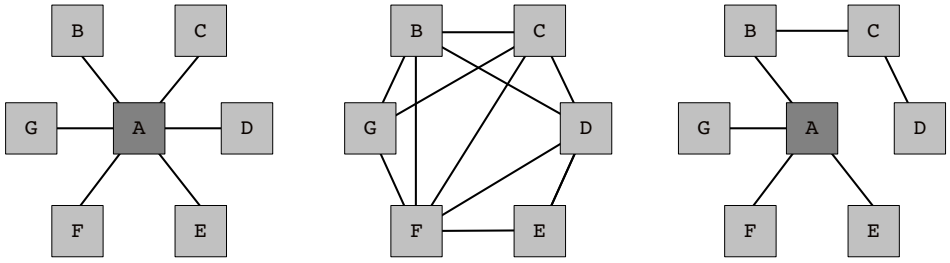


Figure 3.4: Different types of online coupling.

- The centralized approach is recommended for coupling numerous heterogeneous software applications. It can be carried out by client-server architectures, in which one software application has to act as a server, and the others act as clients. Finally, the server provides services that can be used by the clients. The communication is mostly multidirectional (subsection 3.1.2). In civil engineering, centralized online coupling is commonly used to share data between different domains within the planning process or to share functionality (e.g., simulations, calculations) within similar domains.

- The decentralized approach is recommended for the direct coupling of software applications. It can be carried out by peer-to-peer architectures (P2P), in which each software application can act as a client and server simultaneously. Decentralized online coupling enables data and functionality to be shared directly between two software applications without the need for central coordination through a server.

Offline Coupling

Two software applications are offline coupled if the coupling strategy used does not require both software applications to be running. In contrast to online coupling, it does not need to fulfill special preconditions. Therefore, it has become the most widely used coupling strategy for software applications. Offline coupling is achieved through a simple file exchange (Figure 2.5 on page 16). However, software applications use their own, optimized data schemas. The data to be exchanged are exported into external proprietary data schemas (subsection 2.3.1). In order to exchange such data, a conversion, which is also known as mapping, becomes essential (subsection 2.2.3). Offline coupling can be implemented directly or indirectly via an additional standard as shown in Figure 3.5. A qualitative assessment approach of data coupling based on schema mapping is introduced in chapter 6.

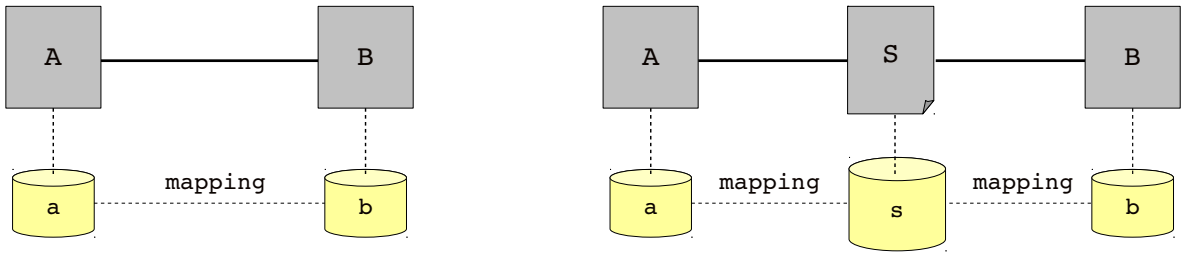


Figure 3.5: Different types of offline coupling.

3.1.5 Shareable Information

In this subsection, the kind of semantic coupling aspect that is considered is one that shares information. A software application has been written to solve a specific task. The software application requires input data, so that it can generate output data, which includes the results. The logical processing of input data to output data, as shown in Figure 3.6, is carried out by interrelated software elements within the software application.

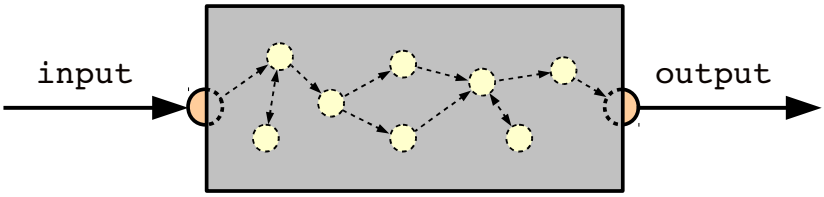


Figure 3.6: Processing of input data to output data.

Due to the fact that software applications are made by companies and organizations, they tend to be more or less proprietary. This means that third-party access to their internal data and functionality may be restricted. Thus, the information to be shared depends on the level of proprietary control. Software can behave similarly to *black-box*, *gray-box* or *white-box systems*.

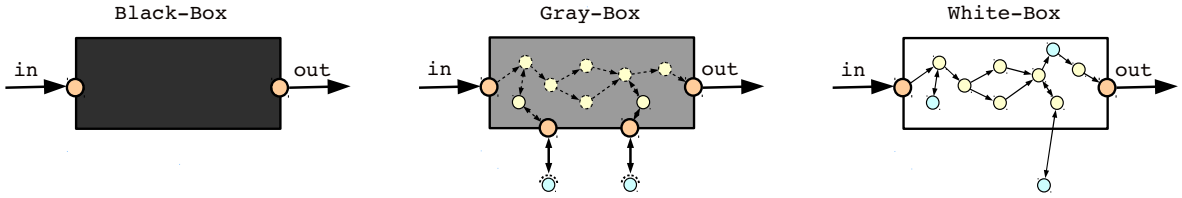


Figure 3.7: Level of proprietary control of software applications.

- A software application behaves like a black-box if it is not possible to influence its internal processing. Black-box software applications are restricted and can only be coupled by data. Errors in the input data can be detected during the import process. Black-box software applications are weak coupled, robust and mostly fail safe.

- A software application behaves like a gray-box if it is possible to influence parts of its internal processing. This can only take place indirectly via defined interfaces. Gray-box software applications are less restricted and can be coupled by data and functionality. The coupling of gray-box applications is stronger than the coupling of black-box applications due to the higher dependencies on internal and runtime processes. Thus, the coupling can be less robust and more prone to errors.

- White-box software applications can also be coupled by data and functionality. They are usually open-source, non-restricted softwares. Coupling occurs directly without any specific interfaces through an adding, replacing, removing of software elements, or a linking of existing functionality externally. Due to the high degree of direct interlacing, this coupling is the strongest, therefore, it can be sensitive and even more error prone.

The coupling of software applications with respect to shareable information can be classified into *data* and *functional coupling*. Both coupling types are important in software coupling. However, data coupling can be applied to each level of proprietary control, whereas, functional coupling can only be applied to the gray-box and white-box level. Due to the fact that the focus of this thesis is on data coupling, it supports the claims for each level of proprietary control. The degree of robustness can be measured by the data exchange examined in chapter 6.

Data Coupling

Data coupling is independent of specific software requirements, therefore, it is widely used for software coupling. The output data of one application is used as input data for the application to be coupled. Data coupling can be combined with other aspects of technical coupling. For example, implementation of an asynchronous offline data coupling can take place in a decentralized manner via a simple file exchange, whereas a synchronous online data coupling can take place centrally via a file sharing server. Due to the different data schemas, the data have to be mapped.

- Due to the restricted access on their internal data schemas, the unidirectional data exchange between two black-box software applications, A and B, results in one data mapping $m_{ab} : a \rightarrow b$. In order to carry out a bidirectional communication process, an additional data mapping $m_{ba} : b \rightarrow a$ is necessary. The coupling of black-box software is mostly achieved by simple file import/export. In the case of gray-box and white-box software, additional data mappings are needed, from the internal data representation of A to the exchange data format, and then to the internal data representation of B.

Figure 3.8: Direct data coupling of black-box software applications.

- The data exchange between two software applications, A and B, via a standard S, results in the following two data mappings for unidirectional communication: $m_{as} : a \rightarrow s$ and $m_{sb} : s \rightarrow b$. In order to implement bidirectional communication, the following two additional data mappings are needed: $m_{bs} : b \rightarrow s$ and $m_{sa} : s \rightarrow a$.

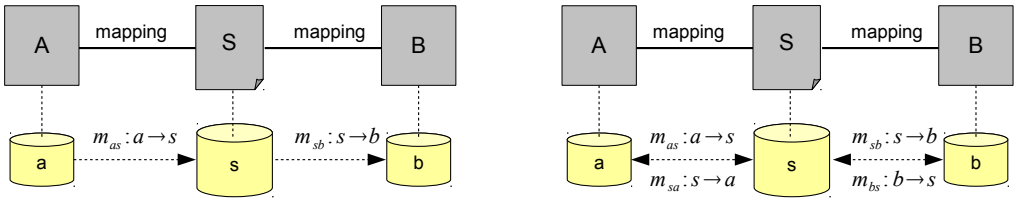


Figure 3.9: Data mappings needed for indirect data coupling.

Functional Coupling

Functional coupling of software applications can be achieved by manipulation mechanisms, which operate on the internal functionality of the applications. This type of coupling is not possible for black-box softwares due to their restricted nature. White-box applications are not restricted and can be manipulated directly in the software application itself. Gray-box software applications, however, are restricted, thereby allowing only indirect manipulation of parts of their internal functionality. Manipulation takes place via defined interfaces and certain software techniques [67], which are listed below:

- Adapters allow single software elements or pieces of software, though incompatible interfaces, to work together. In general, this is enabled by a conversion of interfaces.
- Decorators allow to extend functionality, as well as new behaviors to be independently added to objects at their runtimes.
- Proxies allow to define placeholders for objects that are used to control access, to support distributed objects or to protect components from undue complexity. Any operation on the proxy is delegated to the original object. A proxy can be subdivided into remote, virtual, protection proxy and smart reference.
- In general, software libraries are a collection of resources and functionalities to share and change source codes modular. For the most part, libraries are not executable. Instead they are addressed via references (also known as links), which is usually done by a linker. They can be subdivided into static, dynamic, object, class and remote libraries.
- Commands are used to call methods at a later time. A command has to encapsulate all the information needed, such as method name, the object that owns the method and the parameter values. Commands become powerful in combination with macro languages.

The definition of interfaces can be changed. In addition, the implementation of interfaces in external functions can be bad or wrong. Furthermore, the coupled external function may not be suitable for the original process chain of the software application. All of these aspects can lead to erratic and less robust software behavior. A qualitative assessment of functional coupling can be achieved by white-box and gray-box testing, which is currently an area of extensive research. However, the software environment in civil engineering is mostly characterized by proprietary black-box software systems. Therefore, functional coupling is not examined further in this thesis.

3.2 Coupling Graph

The modeling of couplings depends on the decisions made with regard to coupling aspects (section 3.1) and the software techniques (subsection 2.3.2) used. These decisions are necessary for the successful development, or selection of adequate coupling strategies. Decisions also should take alternative coupling options into account. This can be done with a coupling graph, as shown in Figure 3.10.

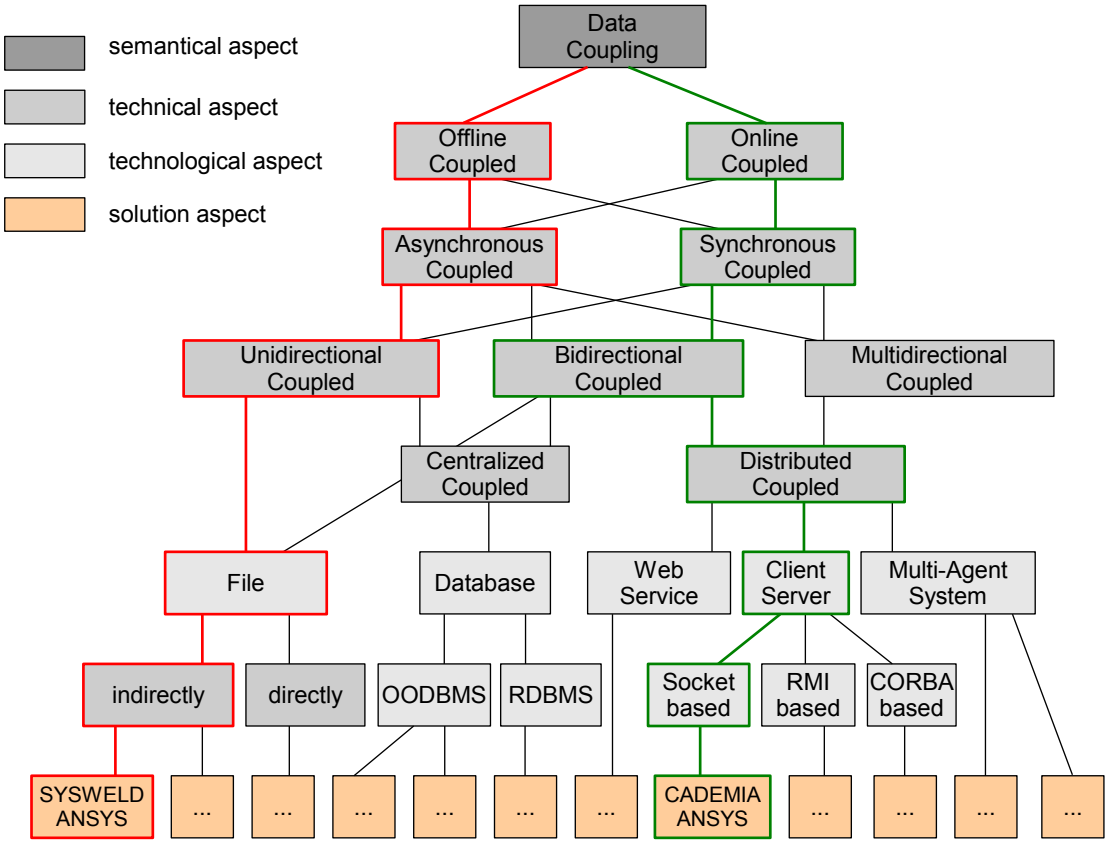


Figure 3.10: Simple coupling graph.

Example: *Two software applications have to be coupled by data coupling. Data coupling can take place offline or online. Offline data coupling can occur asynchronously or synchronously. Asynchronous data coupling can be implemented unidirectionally, bidirectionally, or multidirectionally. Various combinations may exist for each branch, which result in many different coupling strategies. Two data couplings performed in the research training group are shown. The red line represents the decisions that have been made for the coupling between Sysweld and Ansys [147]. The green line represents the coupling strategy between CADEMIA and ANSYS (chapter 5).*

The coupling graph in Figure 3.10 is used to exemplify the different types of decisions, and how they work together to find the right coupling solution. It contains four different kinds of decisions.

- A semantic decision has to be made at the beginning. This determination is followed by the question of what information has to be coupled, which is dealt with in subsection 3.1.5. Semantic decisions are the root elements within the coupling graph, which are represented graphically as a dark gray box.

- Next, various decisions have to be made to address the question of how software applications can be coupled. This includes two different kinds of decisions:
  - The properties of the software coupling that are defined by the coupling aspects (section 3.1), which are represented graphically as a gray box.
  - The existing software techniques to use for performing the communication and data exchange (subsection 2.3.2). They are represented graphically as a light gray box.

  Properties and software techniques can be linked together.

- Finally, various generic coupling solutions are found at the end of the coupling graph. They can contain existing coupling solutions, as well as generic blueprints for future coupling solutions. They are represented graphically as orange boxes and complete a branch in the coupling graph.

It should be noted that this coupling graph is just one of many perspectives. In reality, coupling graphs can be more abstract. Depending on the coupling aspects and software technologies considered, the coupling graphs may have different depths and widths. Coupling graphs should be developed with respect to the skill of the software engineer. They should be as simple as possible. The width and depth of the coupling tree, and hence the number of options provided are inversely proportional to the experience and knowledge of the software engineer. This means that an inexperienced engineer may require more information about the available coupling decisions and software technologies, whereas an experienced engineer usually requires less information in order to find and implement an adequate coupling strategy.

Finally, various levels of abstractions are necessary to successfully describe and classify the multitude of coupling strategies for software coupling. An efficient approach for modeling at different levels of abstraction is given by a metamodel architecture.

3.3 Metamodel Architecture

The model approach is an inherent part in software engineering. It is widely used to describe and solve engineering tasks. Metamodeling refers to the collection of necessary tools (e.g., rules, constraints, terms, and element definitions) that can be used as the grammar for describing domain specific models. A metamodel can be understood as an abstraction of the model itself, therefore a model always conforms to a unique metamodel. Metamodeling creates different levels of abstraction and can be described by multilayered architectures. The objective is to describe a model from a higher and more abstract model layer. This is achieved by the mapping f, which maps a layer M_i to the next higher layer M_{i+1}:

$$\forall\, m \in M_i\ \exists\, n \in M_{i+1} : (m, n) \in f, \quad f : M_i \rightarrow M_{i+1} \tag{3.1}$$

In general, the concept of multilayered architectures is not limited to the number of layers, which means that it would result in an infinite number of abstraction levels. However, this is not feasible. The higher a layer is, the level of abstraction needed is less, therefore closed metamodel architectures are used, in which the top layer conforms to itself. In practice, architectures with four layers have been used, as shown in Figure 3.11. The layer M_0 contains the specific entities of a model, which are defined by their data and/or behavior. The next higher layer M_1 contains the model (e.g., a physical/logical data or process model, or a specific UML or object model) in order to describe the entities in M_0. Layer M_2 contains the metamodel in order to describe the models of M_1. Finally, the top layer M_3 is used to describe the metamodels of M_2 and also to close the architecture by conforming to itself. A brief overview of metamodel architectures currently being used is given in subsection 3.3.1. In subsection 3.3.2, a four-layered meta-coupling architecture is introduced for an abstraction of coupling concepts according to different coupling aspects.

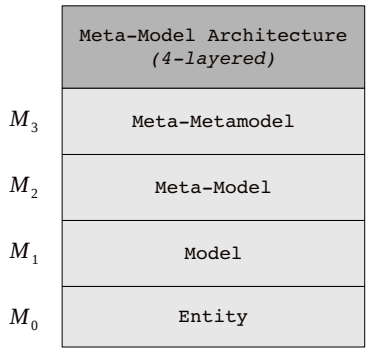


Figure 3.11: A general four-layered metamodel architecture.

3.3.1 Current Metamodels

One of the metamodel architectures currently being used is the Meta Object Facility[2]. It was designed as a four-layered architecture and became an international standard. Important metamodels on layer M_2, based on MOF, are the UML metamodel (ISO/IEC 19501) and the XMI metamodel (ISO/IEC 19503). UML is an inherent part of object-oriented modeling as shown in Figure 3.12 (left side). In civil engineering, metamodel architectures are widely used to define schemas for semantic data exchange, to support particular methods or processes, or to express the additional semantics of existing information. In [57, 152], a four-layered metamodel architecture is used for a consistent concept of modeling products, buildings and knowledge. Another application can be found in building information modeling, which is used for generating and managing building data during its life cycle. This can be achieved by a three-layered architecture as shown with Figure 3.12 (right side). Building instances (layer M_0) are described by the IFC (layer M_1). The IFC data model is an object-oriented data exchange model for building and construction data. The IFC is based on EXPRESS (layer M_2), an object-oriented standard data modeling language for product data. Other standards for metamodeling are ISO 19115 and ISO 15926. ISO 19115 (Geographic information – Metadata) is used for describing geographic information and services. ISO 19115 confirmed metamodels are used for overcoming the huge amount of data needed to provide digital atlases [81] and to identify morphological tendencies [119]. ISO 15926 is located in the process industry and used by various CAD vendors. Unfortunately, there is currently no metamodel architecture for modeling software couplings at different levels of abstraction. Consequently, a four-layered coupling architecture is introduced in the following subsection.

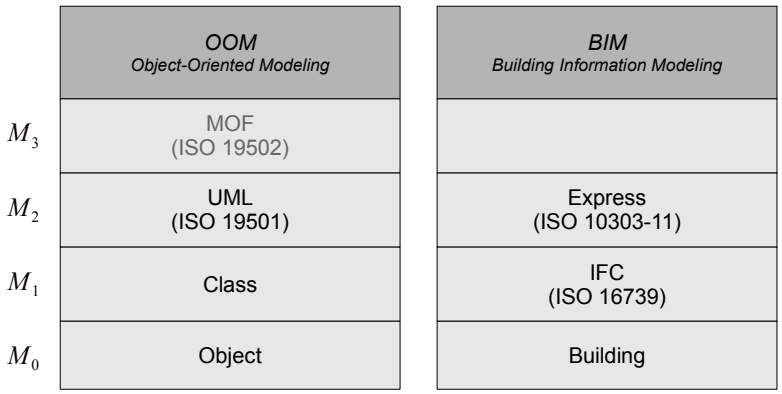


Figure 3.12: Current metamodel architectures.

[2]It is abbreviated as MOF and originated from the Object Management Group (OMG).

3.3.2 Meta Coupling Architecture

The four-layered coupling architecture shown in Figure 3.13 was developed for modeling coupling strategies by using different levels of abstraction. The coupling architecture combines the coupling graph (page 41, Figure 3.10) with the abstraction methods of the metamodel architecture (subsection 3.3) and the design patterns (subsection 2.1.2).

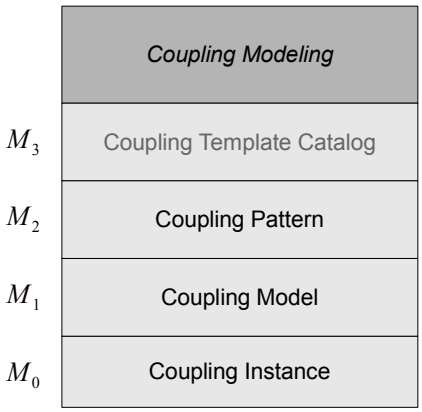


Figure 3.13: Four-layered metamodel coupling architecture.

- The layer M_0 contains the coupling instances of a specific coupling scenario. Instances execute the needed couplings and can vary from being a simple data converter to a complex piece of software. In the object-oriented paradigm, coupling instances are represented by objects. They are implemented in specific programming languages and associated to a special operating system. Coupling implementations on layer M_0 are usually not reusable for similar purposes.

- The layer M_1 describes the coupling instances of M_0 through coupling models. In the object-oriented paradigm, coupling models are related to classes. They define the interactions (relations), the behavior (methods), and the data (attributes). Classes are also implemented in specific programming languages and have to be compiled for different operating systems. Coupling models on layer M_1 are also usually not reusable for similar purposes.

- The layer M_2 describes the coupling models of M_1 through coupling patterns. Coupling patterns are used for modeling coupling graphs and paths. They contain knowledge about coupling aspects and software techniques in order to make decisions during the implementation or the selection process of software coupling. Coupling patterns are documented in a formal way, similar to the well-known design pattern. Thus, they are

independent of specific programming languages and special operating systems. The coupling patterns on layer M_2 are reusable for similar purposes.

- Finally, the layer M_3 describes the coupling patterns of M_2 through a coupling template catalog. The catalog contains various definitions of abstract templates for describing and developing the different types of coupling patterns.

The focus of this thesis is the definition of the coupling template catalog (section 4.1) which is defined in layer M_3, as well as the development of coupling patterns (section 4.2), which is part of layer M_2. The abstract four-layered metamodel coupling architecture in Figure 3.13 is shown in more detail in Figure 3.14.

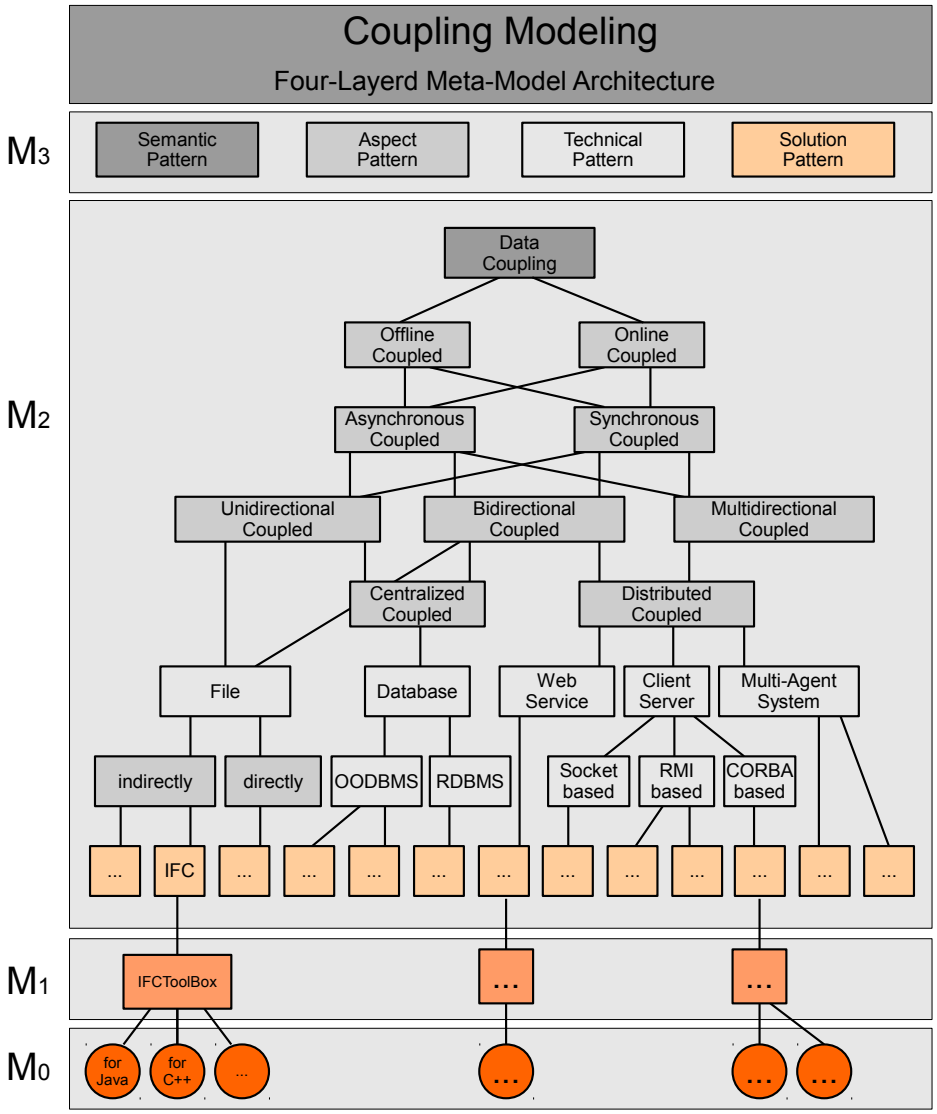


Figure 3.14: Detailed view of the four-layered metamodel architecture.

4 Coupling Pattern Language

4.1 Coupling Catalog

The coupling catalog is located in the top layer M_3 of the meta architecture. It contains four abstract templates for instantiating coupling patterns in layer M_2. Each template corresponds to a specific type of decision within the coupling graph and defines several sections. The sections are used for describing different facts, such as the context in which the pattern is used, as well as advantages and disadvantages and relationships between patterns. However, the development of coupling templates and their sections should not be understood as a final process, but rather as a temporal stepwise process for improving the templates over time.

4.1.1 Semantic Coupling Template

This kind of template covers all of the coupling patterns within the coupling graph, which take the semantic aspects (subsection 3.1.5) into consideration. The coupling patterns, which are based on this kind of template are the root elements of the coupling graph. The template, shown in Figure 4.1 on page 50, is divided into the following three main sections: *context*, *quality assessment* and *collaboration/relationship*.

- Section 1 is strongly related to the template schema of the GoF. It addresses the main context of the pattern. The subsections are as follows: *Name*, *Also Known as*, *Intent*, *Description*, *Applicability*, *Participants* and *Collaboration*.
- Section 2 is related to the assessment of software coupling, which is highly affected by the semantic aspects. Section 2 consists of the following four subsections: *Strategies*, *Requirements*, *Challenges* and *Discussion*. They take assessment strategies into account and mention the requirements and challenges pertaining to the assessment process.
- Section 3 is related to the coupling graph. It consists of the following three subsections: *Alternatives*, *Cooperation*, and *Linked to*. They are used to model the paths, navigate within the graph and describe more complex structural and behavioral properties.

4.1.2 Technical Coupling Template

This kind of template covers all of the coupling patterns within the coupling graph, which take the technical aspects (section 3.1) into consideration. The coupling patterns, which are based on this kind of template are used to model the coupling paths of the coupling graph. Technical patterns define the properties of a coupling strategy. The template, shown in Figure 4.2 on page 51, is divided into the following three main sections: *context*, *technical background* and *collaboration/relationship*.

- Section 1 is strongly related to the template schema of the GoF. It addresses the main context of the pattern. The subsections are as follows: *Name*, *Also Known as*, *Intent*, *Description*, *Applicability*, *Participants* and *Collaboration*.

- Section 2 is related to the technical background in which the pattern is classified. It consist of the following four subsections: *Category*, *Variants*, *Restrictions* and *Discussion*. They take the software engineering aspects into account, outline variants and mention restrictions.

- Section 3 is related to the coupling graph. It consists of the following three subsections: *Alternatives*, *Cooperation*, and *Linked to*. They are used to model the coupling paths, navigate within the coupling graph and describe more complex structural and behavioral properties.

4.1.3 Technological Coupling Template

This kind of template covers all of the coupling patterns within the coupling graph, which take the technological aspects (section 2.3) into consideration. The coupling patterns, which are based on this kind of template are used to model the coupling paths of the coupling graph. Technological patterns describe the software concepts and technologies needed to enable the technical aspects. The template, shown in Figure 4.3 on page 52, is divided into the following three main sections: *context*, *technological background* and *collaboration/relationship*.

- Section 1 is strongly related to the template schema of the GoF. It addresses the main context of the pattern. The subsections are as follows: *Name*, *Also Known as*, *Intent*, *Description*, *Applicability*, *Participants* and *Collaboration*.

- Section 2 is related to the technological background of the software methods and technologies examined. It consist of the following five subsections: *Concept*, *Technical*

Aspect, *Restrictions*, *Examples* and *Discussion*. They describe how to implement technical aspects by the use of an overriding software method. The implementation should be guided through source code fragments and implementation details.

- Section 3 is related to the coupling graph. It consists of the following three subsections: *Alternatives*, *Cooperation*, and *Linked to*. They are used to model the coupling paths, navigate within the coupling graph and describe more complex structural and behavioral properties.

4.1.4 Solution Coupling Template

This kind of template can be found at the end of each coupling path. It includes a generic description of a coupling model, which is defined by its coupling properties and the software concepts used. The coupling properties are represented by technical coupling patterns, whereas the software concepts used are described by the technological coupling patterns of the coupling path. The template, shown in Figure 4.4 on page 53, is divided into the following two main sections: *context* and *coupling model*.

- Section 1 is strongly related to the template schema of the GoF. It addresses the main context of the pattern. The subsections are as follows: *Name*, *Also Known as*, *Intent*, *Description*, *Applicability*, *Participants* and *Collaboration*.

- Section 2 is related to the coupling model. It consists of the following five subsections: *Coupling type*, *Properties*, *Methodology*, *Availability* and *Implementations*. They take the existing generic implementations into account and describe the technical feasibility on the basis of the restrictions arising from the technical and technological coupling patterns.

4.1.5 Template Design

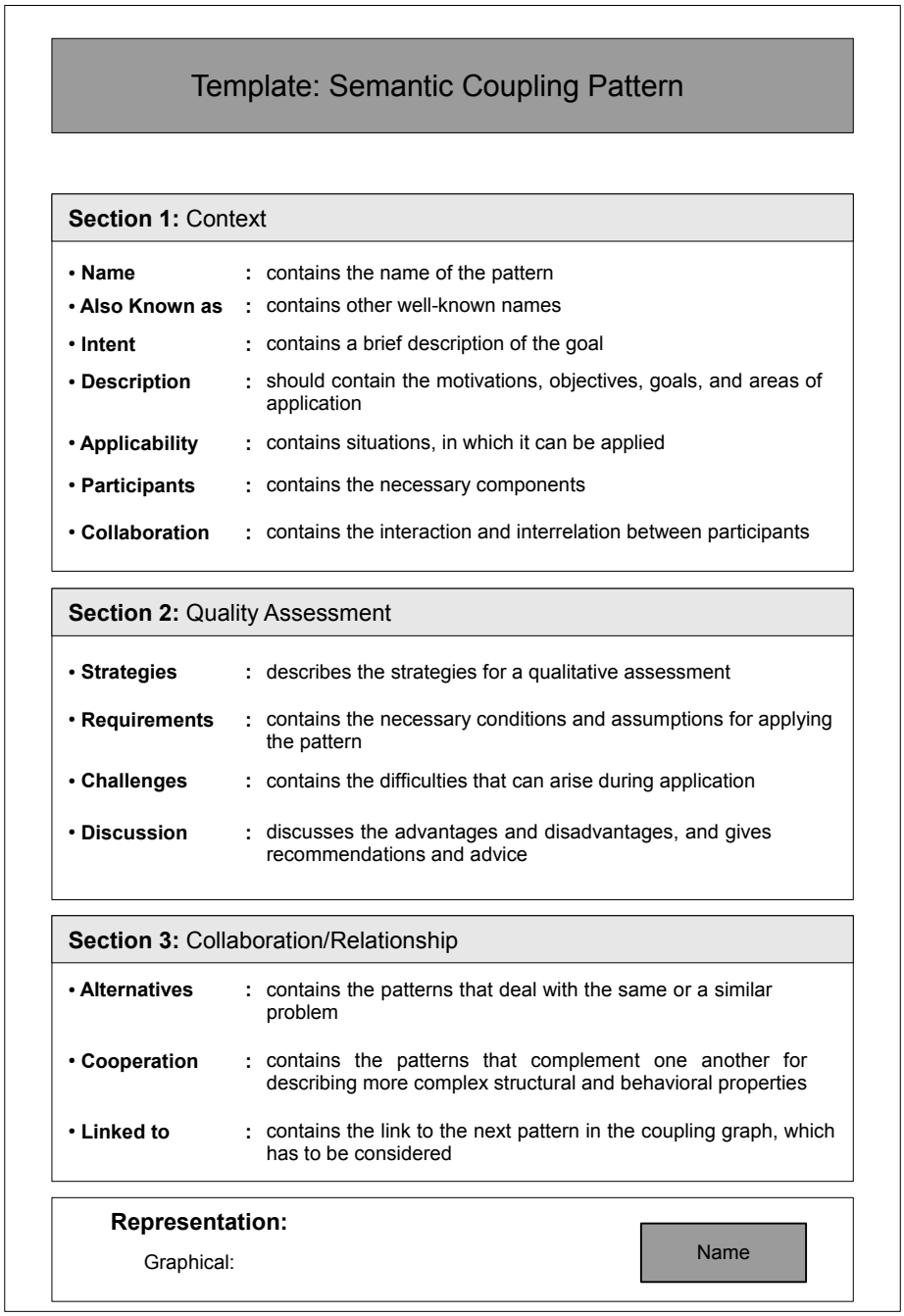


Figure 4.1: Semantic Coupling Template.

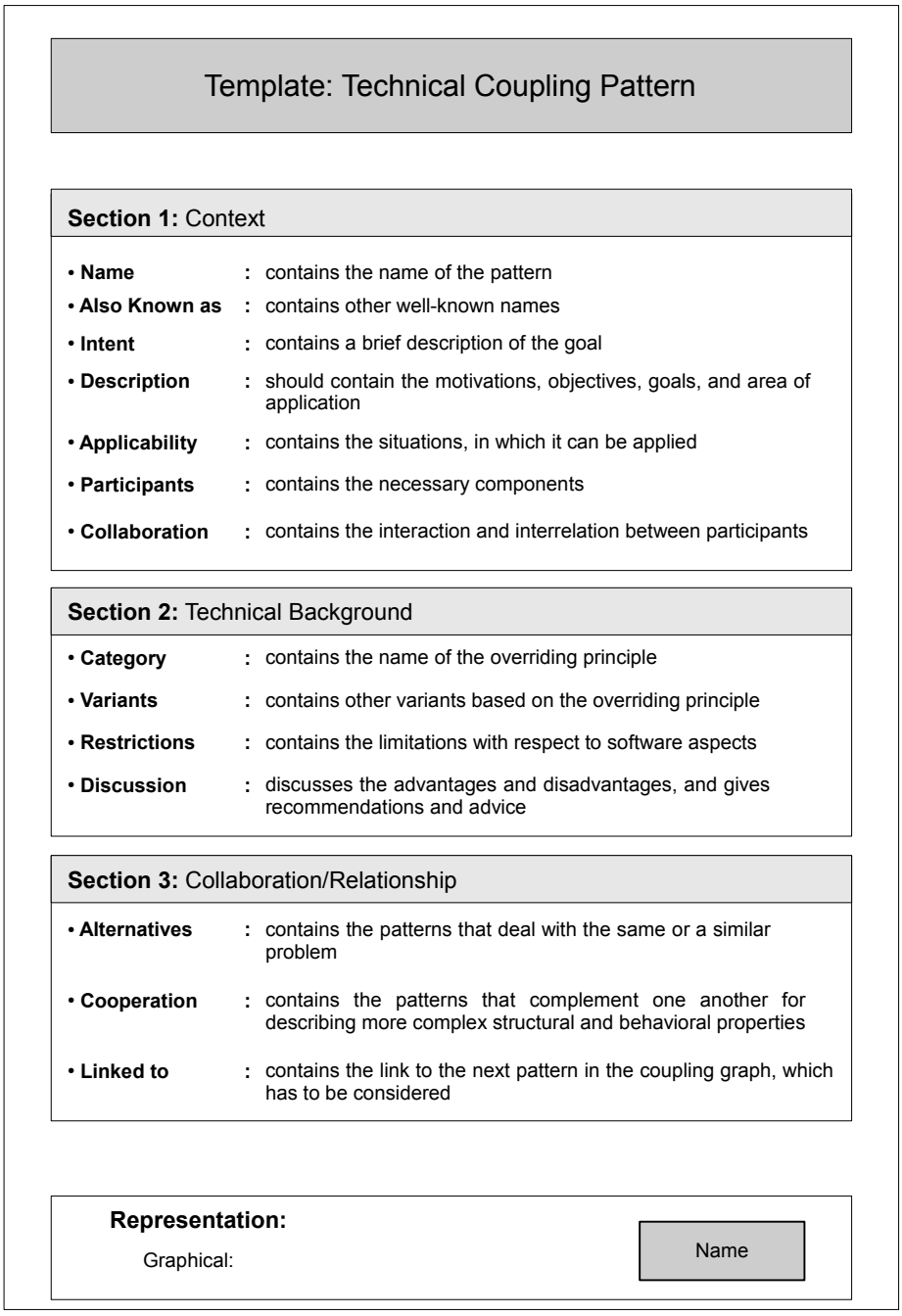


Figure 4.2: Technical Coupling Template.

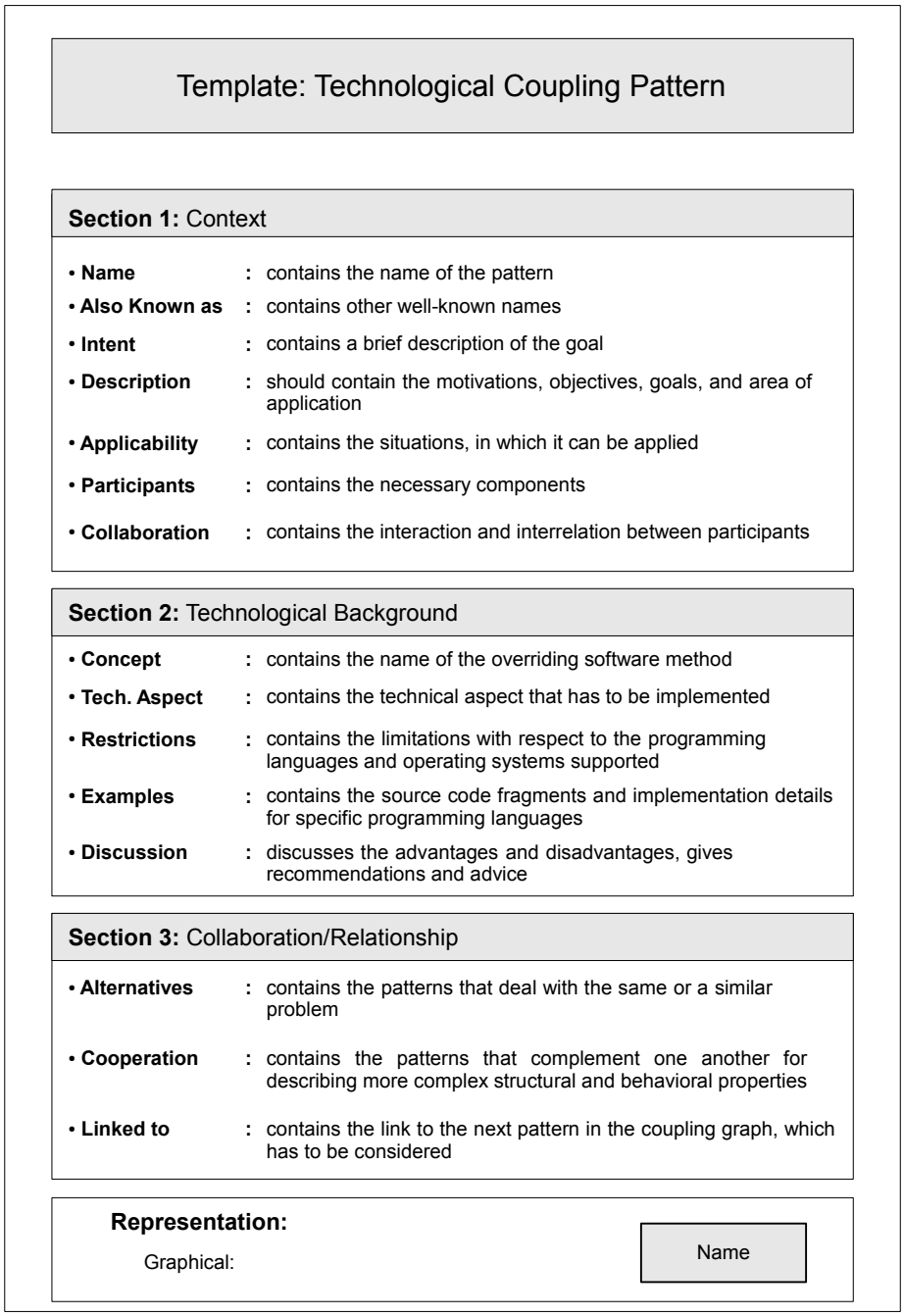


Figure 4.3: Technological Coupling Template.

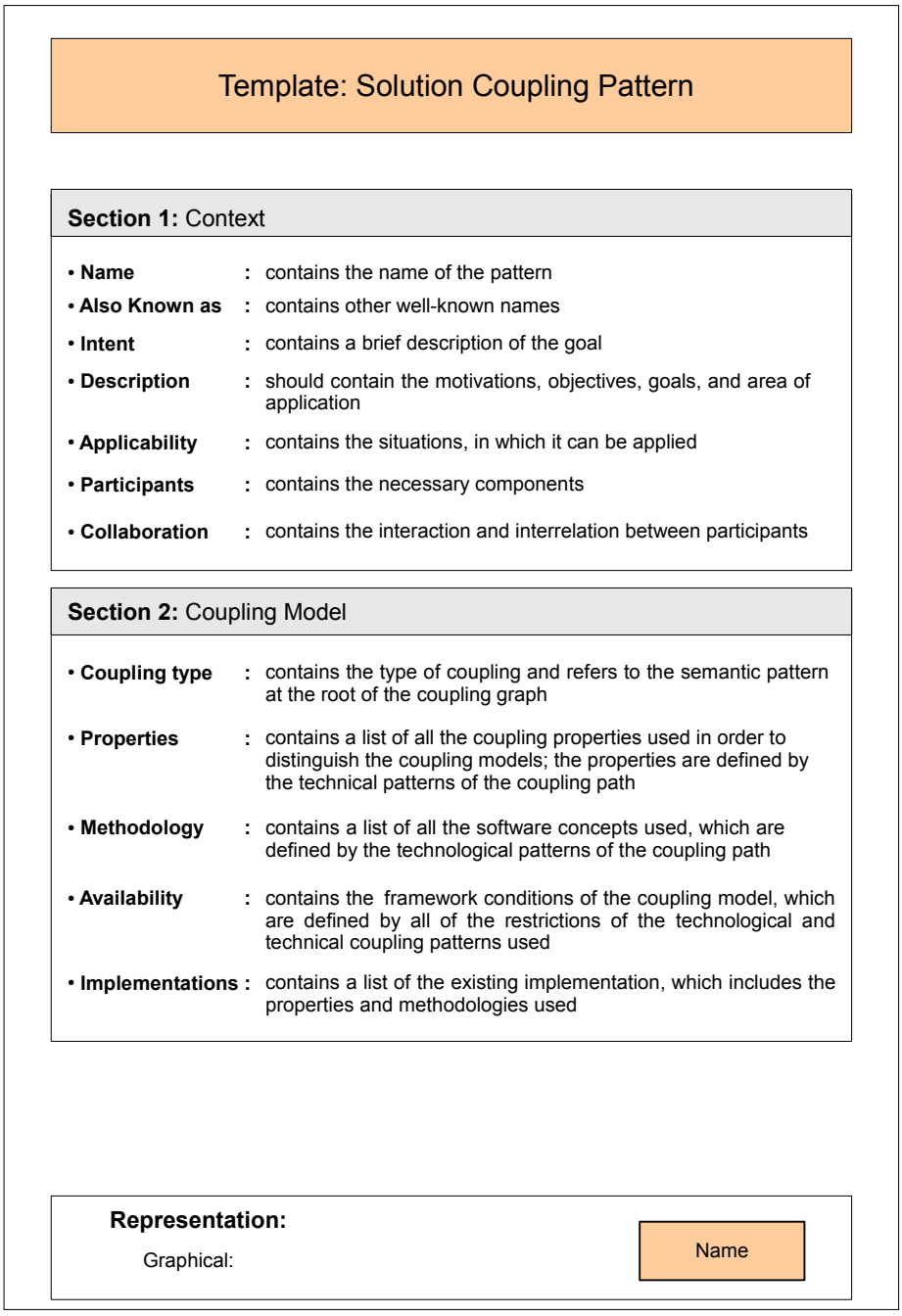


Figure 4.4: Solution Coupling Template.

4.2 Coupling Patterns

The coupling patterns are located in layer M_2 of the meta architecture. They are created according to the coupling pattern templates in layer M_3 (section 4.1) and the specific type of decision within the coupling graph, which is considered. Coupling decisions are important in software development. They have to be taken into account when developing new coupling strategies or selecting adequate coupling strategies from existing ones. This is exemplified by two different types of coupling patterns: the *Data Coupling* and *Client-Server* pattern.

- The *Data Coupling* pattern (subsection 4.2.1) is an instance of the *Semantic Coupling Template* (subsection 4.1.1). This kind of template covers all of the coupling patterns within the coupling graph, which take the semantic aspects (subsection 3.1.5) into consideration. The *Data Coupling* pattern is the root element of the coupling graph. It contains the requirements and challenges for implementing data coupling between software applications.

- The *Client-Server Coupling* pattern (subsection 4.2.2) is an instance of the *Technological Coupling Template* (subsection 4.1.3). This kind of template covers all of the coupling patterns within the coupling graph, which take the technological aspects (section 2.3) into consideration. The *Client-Server Coupling* pattern contains the concepts and restrictions for enabling distributed collaboration.

4.2.1 Data Coupling

Data Coupling

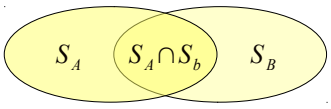
<instance of>

Semantic Coupling Pattern *1/3*

Section 1: Context

- **Name** : Data Coupling
- **Also Known as** : Stamp Coupling, Data-structured Coupling
- **Intent** : Software applications are coupled by data. Data can vary from simple parameters to complex data structures.
- **Description** : Data coupling is widely used in software coupling. The output data of one application is used as the input data for the application to be coupled.

- **Applicability** : Use data coupling when the data included in one software application is needed as a requirement for solving a task in another one.
- **Participants** : Software *A*, Software *B*, data to be exchanged
- **Collaboration** : - Software *A* writes the data to be exchanged in its own data format S_A
  - Sofware *B* reads the data and interprets the data according to its own data format S_B
- **Requirements** : Both data formats (also known as schemas) must have a semantic overlapping $S_A \cap S_B \neq \emptyset$.



- **Challanges** : Data mapping between different schemas.

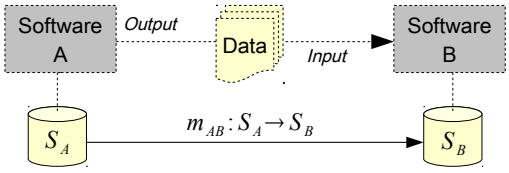


Figure 4.5: Data Coupling Pattern – 1/3.

Data Coupling

<instance of>

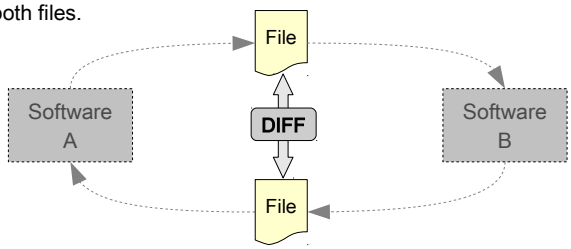
Semantic Coupling Pattern *2/3*

Section 2: Quality Assessment

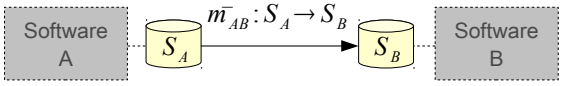
• Strategies :

- **Visual inspection:** The data of software *A* have to be read by software *B*. Differences in the data are then detected optically.

- **File comparison:** The data of software *A* must be written into a file, which then has to be read by software *B*. Software *B* then has to write the data (without any changes) into a second file. Differences in the data are detected by comparing the data in both files.



- Evaluated data/schema mapping: The schemas of software *A* and *B* have to be analyzed according to the overlapping data structures. The data structures found then have to be mapped from schema S_A to schema S_B. The mapping can be evaluated on the attribute level.



• Discussion :

• *Visual inspection* can be done easily. However, it can only detect major problems, like missing or misrepresented entities. It can be used as a first step to assess data coupling roughly.

• *File comparison* instead can detect further minor changes on the data. It can be implemented quickly, but is limited to certain conditions: the exchange must occur via a common data format and the single objects must be identifiable.

• *Evaluated schema mapping* can also detect minor changes on the data. However, an implementation of the mapping and evaluation process is more time-consuming than the file comparison approach but not limited to a common data as well as on identifiable objects. I

Figure 4.6: Data Coupling Pattern – 2/3.

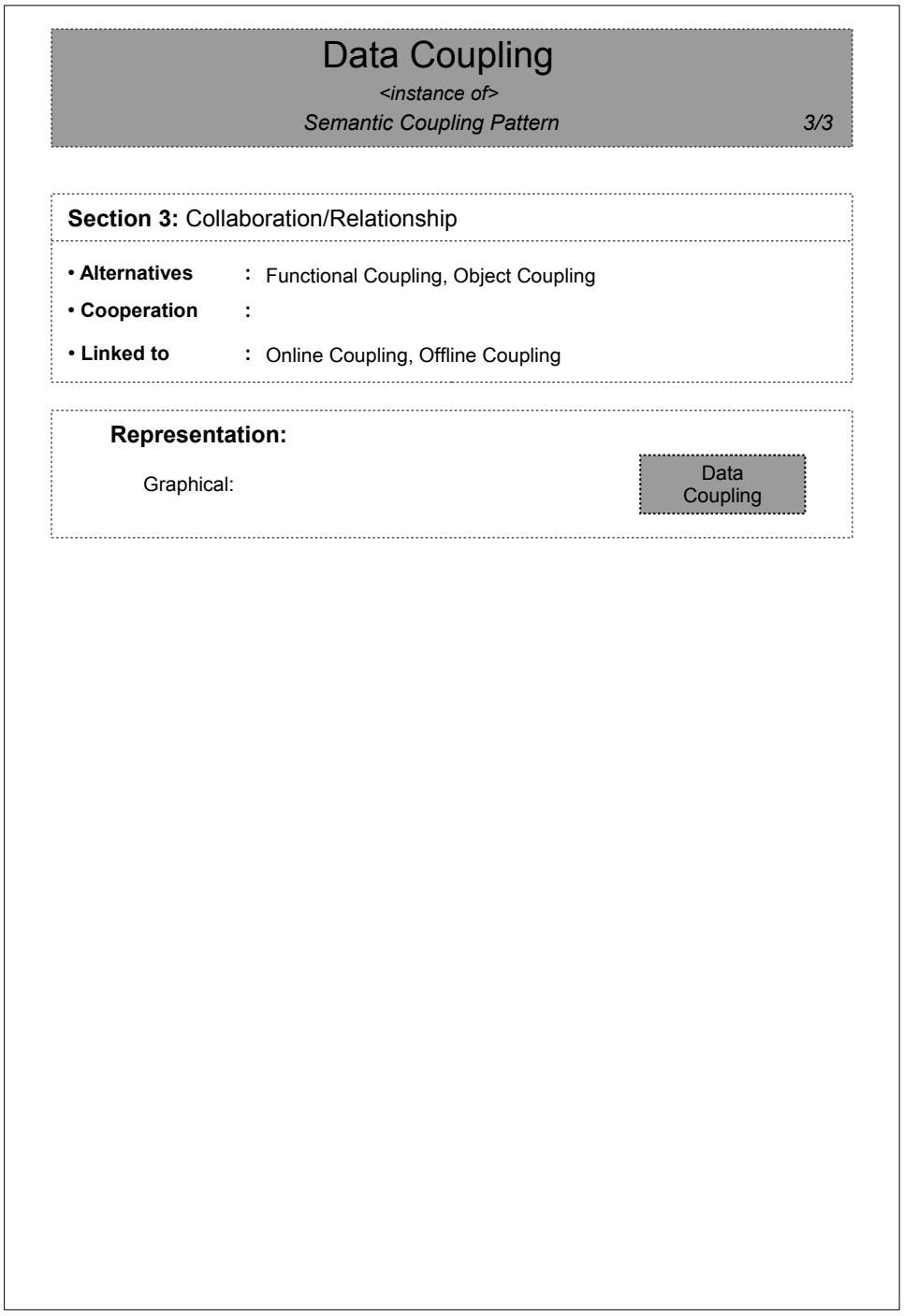


Figure 4.7: Data Coupling Pattern – 3/3.

4.2.2 Client-Server Coupling

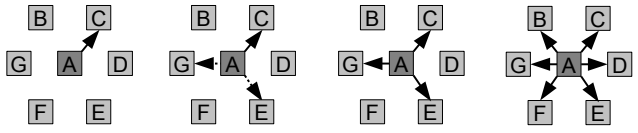
Client-Server

<instance of>

Technological Coupling Pattern *1/3*

Section 1: Context

- **Name** : client-server
- **Also Known as** : client-server architecture, n-tier architecture (two-tier, three-tier)
- **Intent** : the sharing of resources and/or services between locally distributed software applications and different address spaces
- **Description** : • software applications, which share resources and services are called *servers;* a server commonly acts passive and waits for requests
  • software applications, which request a server's resource or service are called *clients;* a client acts active and has to initiate the communication to the server
  • the *communication* between server and clients is enabled mostly by a computer network
- **Applicability** : shareable services are e.g. email exchange, web and database access, simulations, calculations, etc.; shareable resources are e.g. data (files, documents, folders), printers, etc.
- **Participants** : server, clients, communication channel
- **Collaboration** : The server *(A)* provides resources and/or services which can be used by the clients *(B-G)* via ***client request***. The result of the request is sent back to the clients as ***server response***. The communication between server and client is commonly based on the degree of communication in computer networks:



- **Requirements** : network, communication media
- **Challanges** : dependability, performance, simultaneous requests, access privileges, data redundancy, etc.

Figure 4.8: Client-Server Coupling Pattern – 1/3.

Client-Server

<instance of>

Technological Coupling Pattern *2/3*

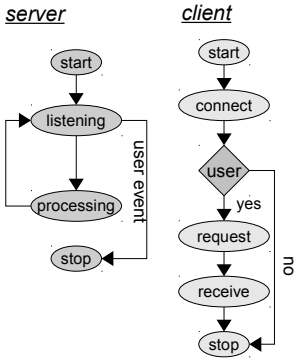
Section 2: Technological Background

- **Concept** : network architecture
- **Tech. Aspect** : distributed coupled
- **Restrictions** : • the implementation of client-server architectures is supported by the most programming languages and operating systems

  • restrictions arise from the used communication protocol → see pattern: *Socket based*, *RMI based*, *CORBA based*
- **Examples** : state/activity diagrams:

Server
• waits for incoming requests (*listening*)
• requests are performed (*processing*)
• results are delivered (*responding*)

Client
• user authentication (*connecting*)
• demand for services (*requesting*)
• waits for server response (*receiving*)



sequence diagrams:

• a single-thread server interrupt the listening in order to process a client request; only one request can simultaneously be processed
• a multi-thread server does not interrupt the listening in order to process a client request; multiple client requests can simultaneously be processed

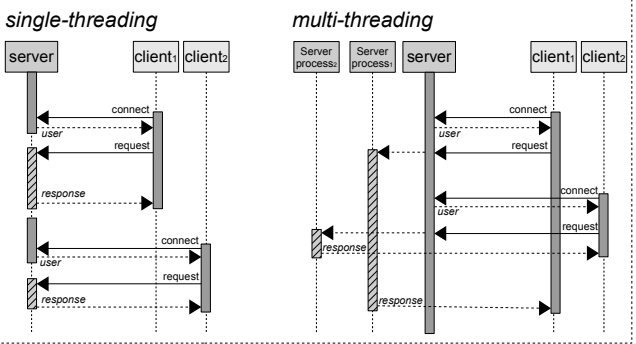


Figure 4.9: Client-Server Coupling Pattern – 2/3.

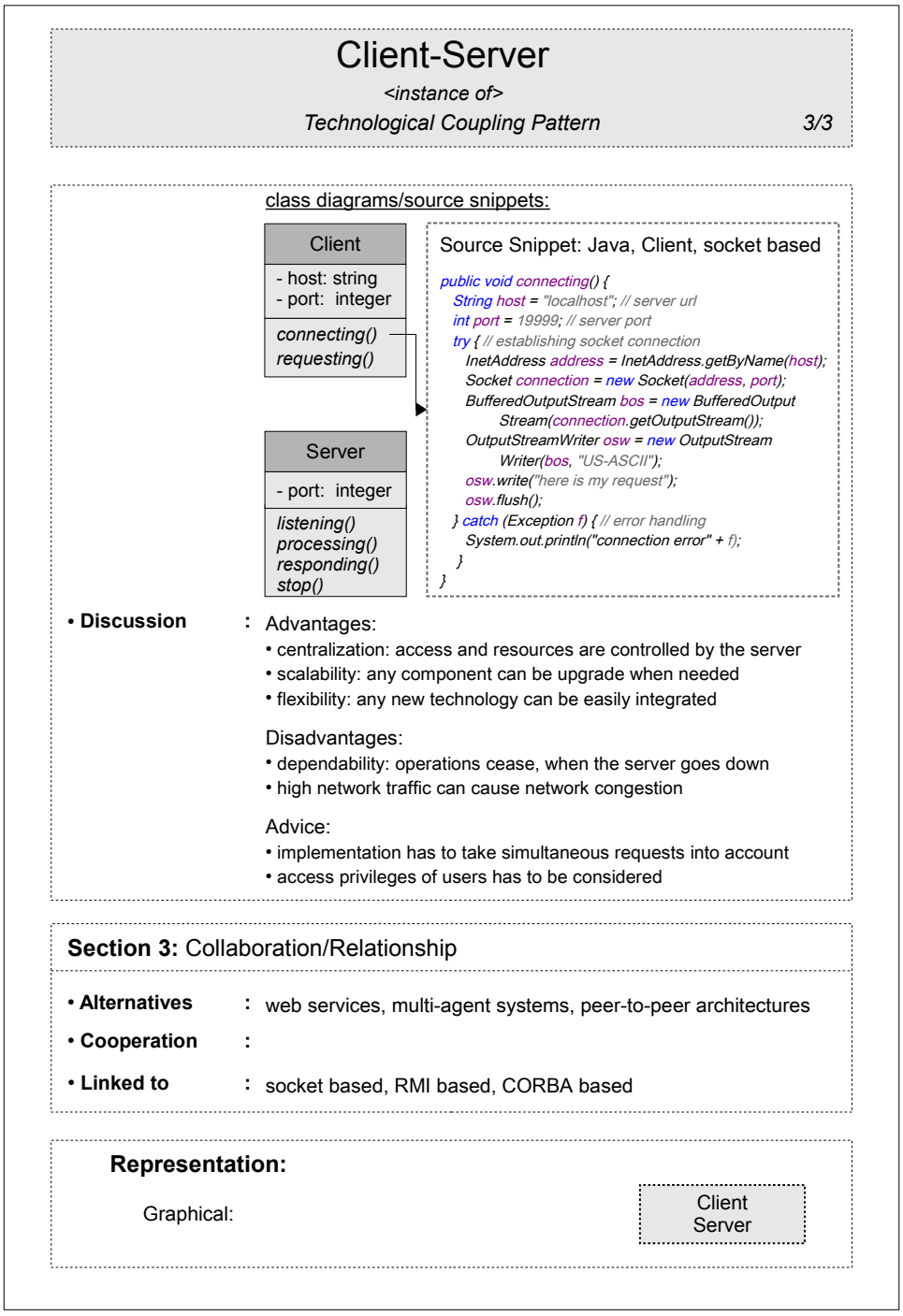


Figure 4.10: Client-Server Coupling Pattern – 3/3.

5 Coupling CAD/FEM

Within the research group, the various sub-tasks have to be examined with the aid of different software applications. Since the tasks are closely intertwined, software coupling is a precondition to be able to work cooperatively and process complex engineering tasks. However, this is complicated due to heterogeneous software environments. In this chapter, the coupling between CAD and FEM software is shown, with particular emphasis on the coupling of CADEMIA and ANSYS in order to combine design and structural analyses.

- ***CADEMIA***, a platform for geometry-oriented AEC applications [56], was chosen to support the design process. It is programmed in Java and based on modular design principles, which offer many possibilities for integrating other functionalities [72]. They can be added through self-developed macros and mouse gestures based on the program's command line language, as well as on self-developed plugins. The plugin system is recommended for integrating new functionalities dynamically at runtime without limitations with regard to the existing functionality.

- ***ANSYS***, a FEM-based engineering simulation software, was chosen to perform the structural analyses. It is widely applied in domains, such as structural and fluid mechanics, acoustics and thermodynamics[11]. It is used in the majority of sub-projects within the research group. The handling, automation and monitoring of the pre/post-process is enabled via commands defined by APDL, its parametric definition language. In addition, sequences of commands can be summarized to macros to perform specific tasks, like geometrical modeling. Furthermore, macros can be nested, which enables the creation of custom commands.

The benefit of linking CADEMIA and ANSYS is being able to combine design purposes with structural analyses. At this stage, the planning process is highly iterative. Structural analyses of an early building design may often lead to a new design, which also has to be analyzed until the building design is finished. This inevitably leads to a great deal of communication between the engineers and consequently between the software applications used. Hence, a synchronous bidirectional online coupling is recommended for handling this iterative process.

5.1 Coupling Strategy

The coupling strategy chosen is *data coupling*, as shown in Figure 5.1. The essential data that comes up during the design process have to be exchanged to perform the structural analyses. The results then have to be transferred back in order to evaluate the design that was created.

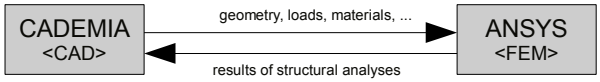


Figure 5.1: Data coupling of CADEMIA and ANSYS.

The information flow has to occur in both directions (*bidirectional coupled*) and in a chronological order (*synchronous coupled*). The chronological order can be achieved by synchronous messaging where the software application waits for a message response before it continues processing. Furthermore, they are running distributed on different computers (*distributed coupled*). Hence, a *client-server* architecture is the proposed strategy to enable distributed collaboration. Due to the fact that both software applications have been developed by different programming languages, a *socket-based* communication is recommended. The corresponding branch with respect to the coupling graph of section 3.2 is shown in Figure 5.2.

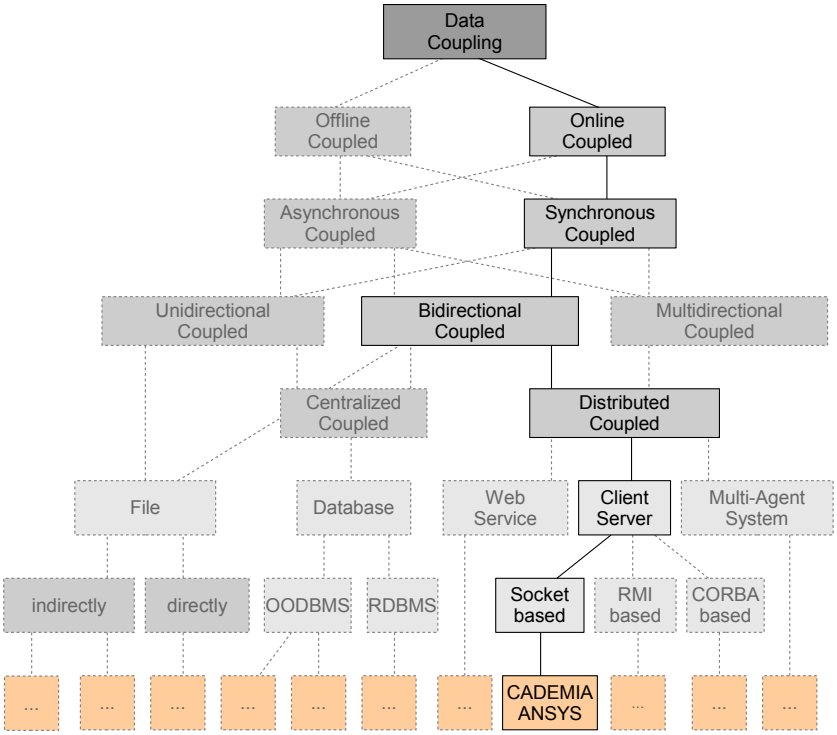


Figure 5.2: Coupling path for coupling CADEMIA and ANSYS.

5.2 Implementation

Technical patterns of the proposed coupling strategy are *online coupled*, *synchronous coupled*, *bidirectional coupled* and *distributed coupled*. The technological patterns arising from these are *client-server* and *socket-based*. They include the following implementation proposals:

- CADEMIA has to act *actively* as a client application. External services provided by the server application can be used via client requests. The results of performed services are included in the server response.
- ANSYS has to act *passively* as a server application. It gets activated only by incoming client requests, which are then processed. The request must contain all of the data needed to perform the service. The results are returned to the client as a server response.
- Due to the different programming languages, it is recommended that communication between the server and clients is done via sockets. The chronological order can be achieved by synchronous messaging where the client waits for a server response before it continues processing.

From a software engineering point of view, CADEMIA and ANSYS provide various ways of integrating and extending additional functionalities.

- The recommended way of implementing the client is via a self-developed CADEMIA plugin, which has to be implemented in Java.
- The recommended way of implementing the server is via a self-developed ANSYS command, which has to be implemented in C.

The final strategy for coupling CADEMIA and ANSYS is shown in Figure 5.3. It takes the different proposals arising from the technical patterns into account, as well as the capabilities of software environments.

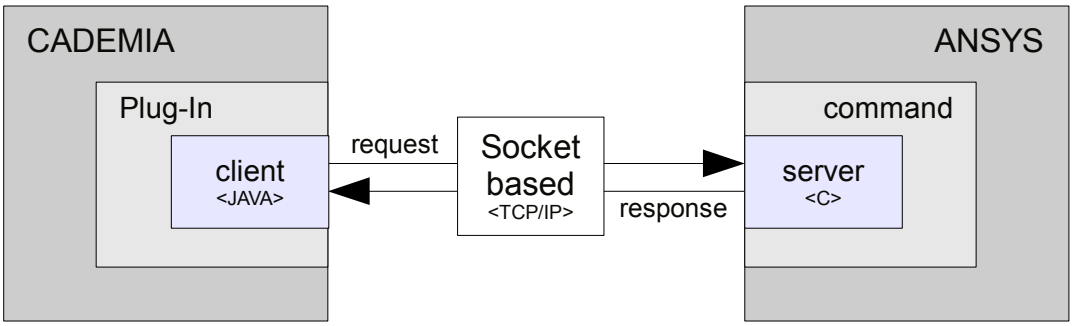
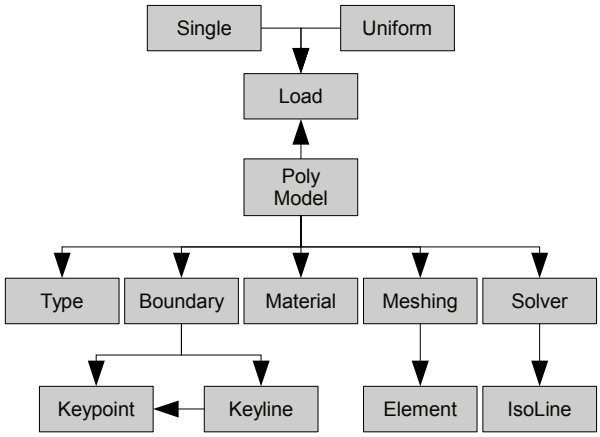


Figure 5.3: Strategy for coupling CADEMIA and ANSYS.

CADEMIA – Client PlugIn

The client plugin shown in Figure 5.4 has been structured into five packages. Three of them are required by CADEMIA's plugin system to extend new functionality, enable user interaction and for visualization purposes. The actual implementation of the client is included in the *mdl* and *cnct* packages, which are explained below:

- The *mdl* package contains a self-developed data schema on the basis of object-oriented principles for managing FEM data. It is essential for modeling FEM aspects within CADEMIA and allowing data exchange with ANSYS.



- The *cnct* package contains components required for achieving network connectivity. The core component, *AnsysConnection*, is implemented based on the *socket-based* pattern. It includes methods for establishing connectivity and sending/receiving data.

Finally, the self-developed plugin allows classical design elements to be combined (supported by CADEMIA) with structural analyses (performed by ANSYS).

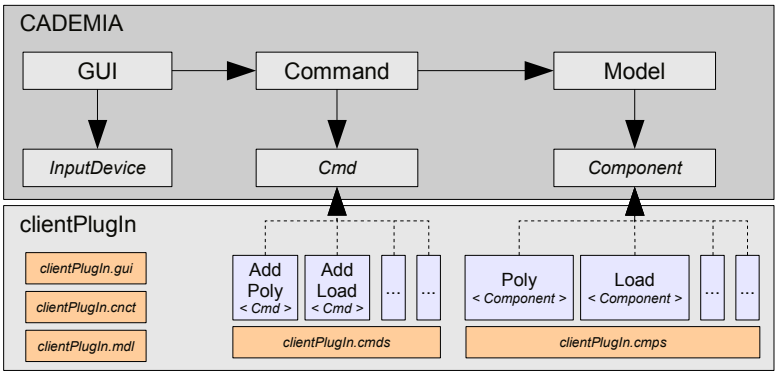


Figure 5.4: The CADEMIA client plugin for linking ANSYS.

ANSYS – Server Command

In order to perform a structural analysis within ANSYS, the following three global steps are essential: the modeling step (which includes geometry, material and mesh generating), the solution step (which includes loads and boundary conditions) and the evaluation and visualization step. However, within the proposed coupling strategy, most of the steps are performed externally by CADEMIA; only the meshing and structural analysis have to be performed by ANSYS. In order to provide these functionalities, a server component has been developed, which is based on the *client-server* and the *socket-based* pattern. It includes the following:

- Receiving external client data by listening on specific network ports
- Processing and analyzing incoming data via ANSYS
- Response results and closing the connection

The communication and interaction between the server component and ANSYS is enabled by its API. The following two functions are of particular importance:

- The *cAnsSendCommand* function allows the server component to execute arbitrary APDL commands within ANSYS at runtime. They are used for modeling geometry, material, loads, etc. The integer value returned indicates whether an APDL command has been executed or not.

```
int cAnsSendCommand(char *strCmd)
```

- The *cAnsGetValue* function is equivalent to the **get* APDL-command. It is used by the server component for querying the results of the structural analyses performed, which are essential for evaluating the building designs of clients. The integer value returned indicates whether a query was successful or not.

```
int cAnsGetValue(char *strGet, double *dbValue, char *strMsg, int *intType)
```

After starting the server component through its defined command, ANSYS acts as a server application. It listens on a defined network port for incoming client data as a basis for performing the structural analyses. Finally, the results of the structural analyses are then returned to the client. Due to the lack of multi-thread support, only one client can be connected at the same time.

5.3 Example of Use

An existing two-dimensional drawing of a two-story building design was imported into the CADEMIA software as shown in Figure 5.5. During the planning process, it was examined for stability with respect to the subsoil.

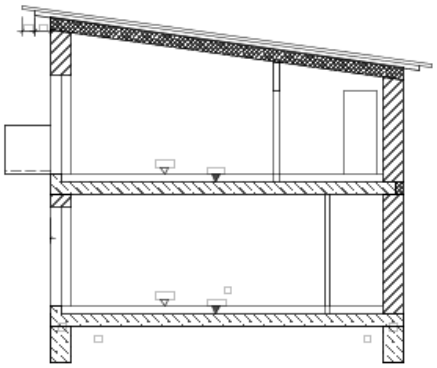


Figure 5.5: Two-story building design.

For this reason, the building design was extended and remodeled by the planning engineer. This was done with the CADEMIA client plugin and its commands, which includes the geometry of the subsoil and its material properties as shown in Figure 5.6.

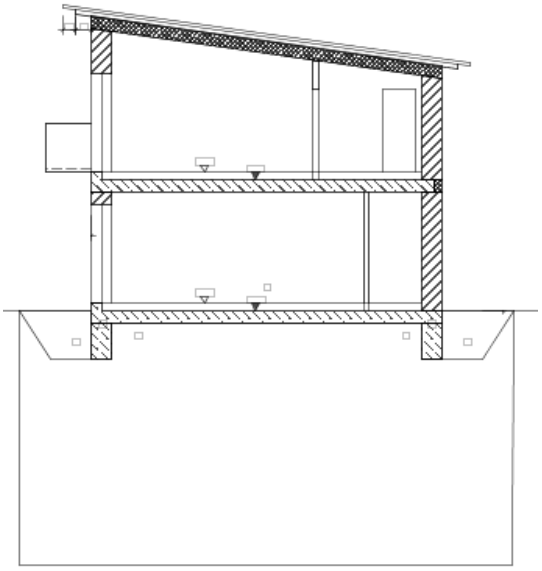


Figure 5.6: Two-story building design with subsoil.

In order to perform a stability analyses, the subsoil first has to be decomposed/meshed into finite elements. However, this is not part of CADEMIA's functionality, therefore it is done externally by ANSYS. Hence, the geometry of the subsoil is transferred to ANSYS (via a socket-based communication channel), which runs as a server application. The incoming geometry data are then meshed as shown in Figure 5.7 (left) and the results of the meshing are returned to CADEMIA as shown in Figure 5.7 (right).

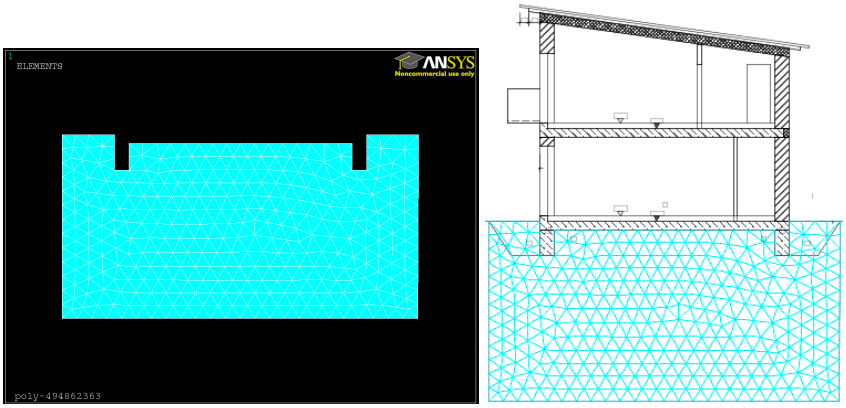


Figure 5.7: The meshing by ANSYS (left) and the results in CADEMIA (right).

In the final step, the loads and boundary conditions have to be modeled by the planning engineer. This occurs through the CADEMIA client plugin and its commands and includes distributed loads for the foundation slab, strip foundations and boundary conditions for the soil body. Based on this, ANSYS can perform a structural analysis as shown in Figure 5.8 (left). Finally, the results of the analysis are returned to CADEMIA as shown in Figure 5.8 (right), which can then be evaluated by the planning engineer.

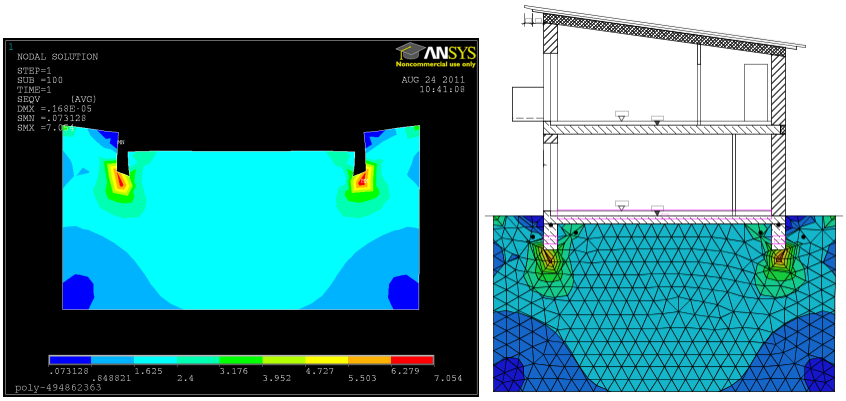


Figure 5.8: The analysis by ANSYS (left) and the results in CADEMIA (right).

6 Evaluated Data Coupling

Data coupling (subsection 3.1.5) is a root element of the coupling graph and thus the basis for a multitude of coupling strategies (section 3.2). However, perfect semantic interoperability cannot be expected (subsection 2.3.1). For this reason, a generic assessment approach based on schema mapping is introduced (section 6.1). It is exemplified by a typical coupling scenario from civil engineering (section 6.2). The mathematical formulation (section 6.3) consists of the schema mapping process and its assessment. Uncertainties within the schema mapping process are considered (section 6.4). Finally, the quality of the data exchange and hence the quality of the coupling can be expressed by a global value.

6.1 Solution Approach

The assessment approach depends on the participating schemas and their mappings. This is exemplified in Figure 6.1. The participating data schemas are a, b and standard S:

$$a := \{a_1, a_2, a_3\};\ S := \{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8\};\ b := \{b_1, b_2, b_3\}$$

The corresponding schema mappings used for the unidirectional exchange of data from a to b via standard S are $m_{aS} : a \to S$ and $m_{Sb} : S \to b$. Each mapping within m_{aS} and m_{Sb} must be disjoint: $m_{aS} = \{m_{aS_1} \cup m_{aS_2} \cup m_{aS_3}\};\ m_{Sb} = \{m_{Sb_1} \cup m_{Sb_2} \cup m_{Sb_3}\}$.

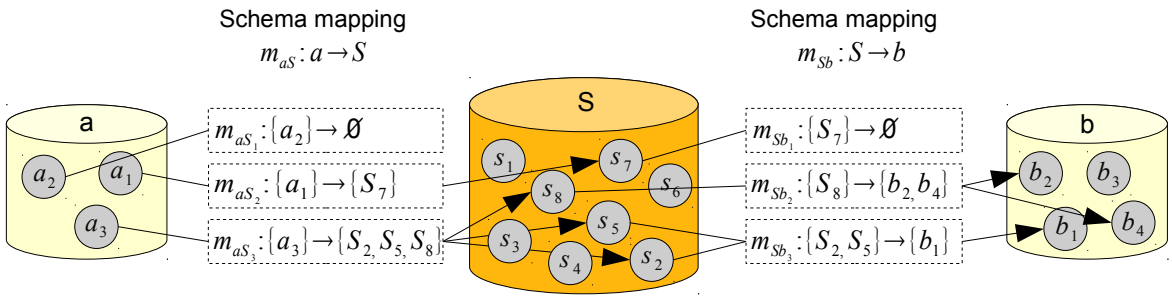


Figure 6.1: Mapping of data structures.

- Data structure $a_2 \in a$ cannot be mapped to data schema S because there are no equivalent data structures. Therefore, a_2 is not exchangeable to S; and consequently it is also not exchangeable to schema b.
- Data structure $a_1 \in a$ can be mapped via m_{aS_2} to data structure $S_7 \in S$. However, S_7 cannot be mapped to schema b because there are no equivalent data structures. Therefore, a_1 is not exchangeable to data schema b.
- Data structure $a_3 \in a$ can be mapped via m_{aS_3} to data structures $\{S_2, S_5, S_8\} \subseteq S$ and via m_{Sb_2} and m_{Sb_3} to data structures in b. Therefore, a_3 is exchangeable to schema b.

An assessment of data structure mapping can be done on the basis of the attributes of each data structure. The resulting quality values (subsection 6.3.3) describe the quality level of the mapping of the corresponding data structures. Their domain can be arbitrarily defined either linguistically or numerically. Finally, the overall coupling quality for a given set of exchangeable instances can be computed a priori.

6.2 Scenario

A typical scenario from civil engineering is used to illustrate the formalization process of schema mapping (subsection 6.3.2) and the assessment process (subsection 6.3.3).

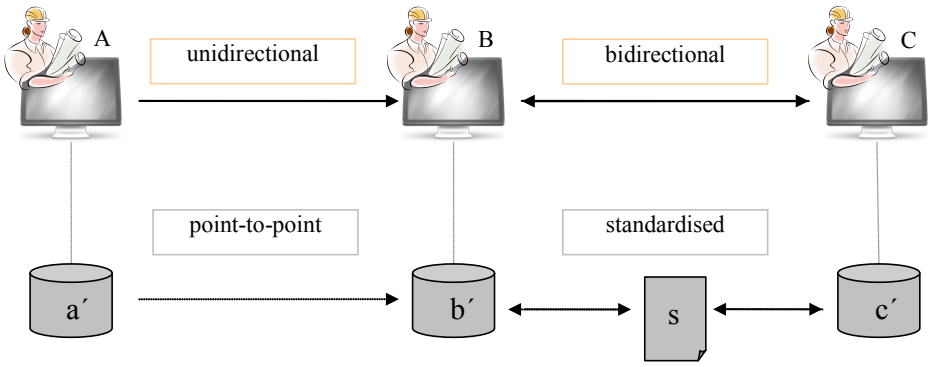


Figure 6.2: A typical coupling scenario from civil engineering.

Coupling scenario: In the early planning phase, architect *A* makes a preliminary design of the building. Engineer *B* must then carry out a complex structural analysis. Thus, the existing planning data have to be exchanged *unidirectionally*. Due to the complex structural analysis, *B* has to collaborate with an additional engineer *C*. The data exchange is *bidirectional*. Due

to the numerous software applications on the market, different applications and schemas are involved within the planning process. Engineers *B* and *C* finally agree on an exchange of planning data through an additional *standard s*.

6.3 Formalism

6.3.1 Basics of Data Coupling

Two software applications are said to be data structure coupled if they exchange data via a schema. During the planning process many software applications with different neutral or proprietary schemas are to be coupled. In this context, set *Q* contains all the schemas (Eq. 6.1) and set *C* all the schema couplings (Eq. 6.2), both of which are needed to solve the task.

$$Q := \{q | q \text{ is a schema}\} \tag{6.1}$$

$$C := \{(a, b) \in Q \times Q | \text{ Schema } a \text{ is to be coupled with schema } b\} \tag{6.2}$$

Data structure coupling can be implemented in two different ways. The *point-to-point* concept couples two schemas $a, b \in Q$ of two different applications directly. The resulting schema coupling is $(a, b) \in C$. The exchangeable information is described by $a \cap b$. A coupling of *m* applications in this way leads to *n* schema couplings, which is expressed as follows:

$$n = m \cdot (m - 1) \tag{6.3}$$

The standardized implementation instead couples two schemas $a, b \in Q$, of two different applications indirectly via an additional schema $s \in Q$, usually a standard schema. This is reflected in the schema couplings $(a, s) \in C$ and $(s, b) \in C$. The exchangeable information is described by $a \cap s \cap b$. The interrelation between the number of schema couplings *n* and the number of coupled applications *m* is now linear:

$$n = 2 \cdot m \tag{6.4}$$

In data structure coupling, a loss of data or meaning can hardly be avoided (subsection 2.3.1). A misinterpretation of data and information can lead to numerous design errors within the planning process. For this reason, adequate assessment strategies have to be developed. They can be used by designers and engineers as a tool to estimate the reliability of data exchange. The assessment strategy presented in this thesis is based on the evaluation of schema mapping.

6.3.2 Schema Mapping

Schema mapping is used in applications that involve data sharing or data transformation. It plays a central role in data exchange and data integration between heterogeneous software applications. A schema consists of data structures to describe data. A coupling of two schemas can be achieved through the mapping of their single data structures. Mapping templates are shown in Figure 6.3. The deletion pattern is used if there is no equivalent data structure in the target schema. The copy pattern is used if a single data structure in the source schema corresponds to a single data structure in the target schema. The splitting pattern is used if there are multiple data structures in the target schema. The combining pattern is used if more than one data structure in the source schema relates to a single data structure in the target schema. The multiple mapping pattern is used if more than one data structure in the source schema corresponds to more than one data structure in the target schema.

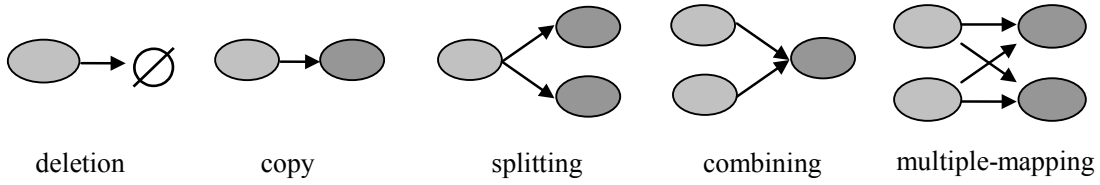


Figure 6.3: Mapping patterns according to Katranuschkov [136].

A schema S is the set of essential data structures to describe and solve a given task:

$$S := \{e | e \text{ is a data structure}\} \in Q \tag{6.5}$$

The data structures inside of a schema can be interrelated to describe more complex facts. These relationships can be formulated with the power set $P(S)$. The power set $P(S)$ of any set S is the set of all subsets of S, including the empty set and S itself. The power set of S contains $|P(S)| = 2^n$ elements, in which n is the number of elements in S.

$$P(S) : \text{Power set of schema } S \in Q \tag{6.6}$$

Finally, the schema mapping of two schemas A and B can be achieved by an unevaluated mapping m_{AB}. It describes the mapping of elements $a \in P(A)$ within schema A to the corresponding elements $b \in P(B)$ wihtin schema B, which is expressed as follows:

$$m_{AB} : P(A) \rightarrow P(B) \text{ with } A, B \in Q \tag{6.7}$$

Example: The mapping m_{AB} of two schemas A and B is shown in Figure 6.4. The templates used are copy (1), splitting (2), combining (3), deletion (4), and multiple mapping (5):

$$A := \{a_1, a_2, a_3\} \in Q \; ; \quad B := \{b_1, b_2, b_3\} \in Q$$

$$m_{AB} := \{(\{a_1\}, \{b_1\}), (\{a_2\}, \{b_1, b_2\}), \ldots, (\emptyset, \emptyset)\}$$

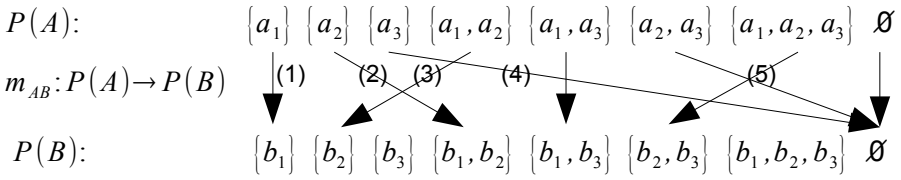


Figure 6.4: Exemplified schema mapping of schema A with schema B.

Application to the scenario example

The scenario example from section 6.2 contains four schemas and five schema couplings, i.e., five schema mappings have to be made to solve the given task:

$$Q = \{a', b', c', s\}; \; C = \{(a', b'), (b', s), (s, b'), (s, c'), (c', s)\}$$

The schema mapping is shown for the schema pair $(c', s) \in C$. The schemas c' and s consists of the following data structures:

$$c' = \begin{Bmatrix} c_1: & \text{myWall3D} \\ c_2: & \text{myMaterial} \\ c_3: & \text{myPoint3D} \end{Bmatrix} \qquad s = \begin{Bmatrix} s_1: & \text{Wall2D} \\ s_2: & \text{Point2D} \end{Bmatrix}$$

The data structures shown in Figure 6.5 are defined by the following attributes: l, w and h are the wall dimensions length, width and height; x, y and z are Cartesian coordinates; e and k are material properties (Young's modulus, heat conductivity coefficient).

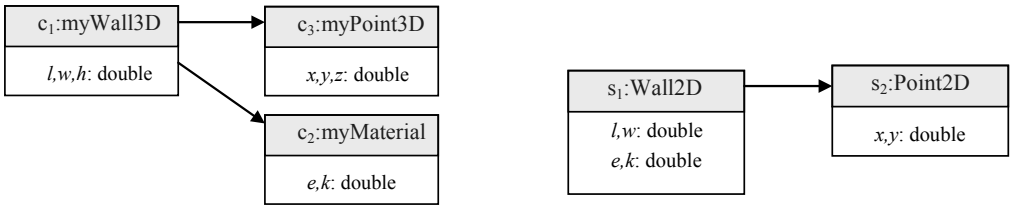


Figure 6.5: Data structures of schema c' and s in UML notation.

Example: The mapping $m_{c',s}$ from schema c' to schema s and the corresponding mapping patterns are shown in Figure 6.6. A wall ($c_1 : myWall3D$) associated with a position point ($c_3 : myPoint3D$) and a material property ($c_2 : myMaterial$) is represented by $\{c_1, c_2, c_3\} \in P(c')$. The corresponding representation in schema s is $\{s_1, s_2\} \in P(s)$ since the data structure $s_1 : Wall2D$ already includes the material properties. Both representations are coupled by the multiple mapping pattern. Up to now, the mapping has not been evaluated. A method for an a priori assessment of the data exchange is described in the next section.

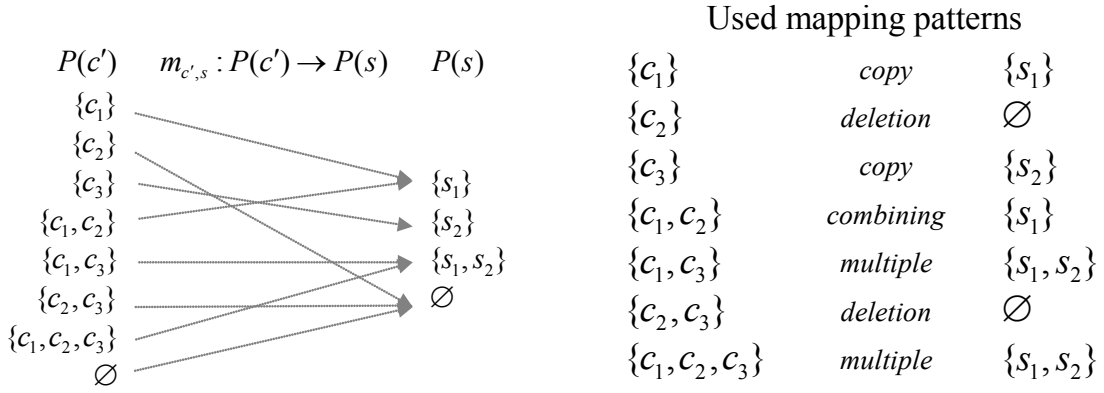


Figure 6.6: Schema mapping and mapping patterns for $m_{c',s}$.

6.3.3 Evaluated Schema Mapping

A set of quality values R is used to assess the data exchange. A quality value $r \in R$ describes the quality of the mapping of two corresponding data structures. The domain can be arbitrarily defined either linguistically or numerically. For now, R is the subset of real numbers $\mathbb{R}$ in the interval $]0, 1]$. A value of 1 means that there is no loss of information during the exchange, whereas a value of less than 1 means that only a part of the information is exchanged. Intermediate quality values are useful for describing the quality of mappings between different mathematical representations (subsection 6.4.4). A value of 0 means a total loss of information, which is represented by the deletion pattern.

$$R := \{r \in \mathbb{R} \mid 0 < r \leq 1\} \tag{6.8}$$

The assessment of data exchange between two schemas $A, B \in Q$ is achieved by the evaluated schema mapping $\bar{m}_{AB}$. Each mapping $(a, b) \in m_{A,B}$ is associated with a value $r \in R$ as follows:

$$\bar{m}_{AB} : m_{AB} \rightarrow R \tag{6.9}$$

So far, the quality of data exchange is defined on the schema level. However, the quality for a specific set of instances has to be computed on the instance level. The prerequisite for doing this is that the data instances must be known. They are included in the set I_A as follows:

$$I_A := \{i \mid i \text{ is a instance of a data structure } e \in A \in Q\} \tag{6.10}$$

The relationships of instances $i \in I_A$ follow the relationships defined in the underlying schema $A \in Q$ exactly. Consequently, the relationship between an instance and its schema is:

$$type : P(I_A) \rightarrow P(A) \tag{6.11}$$

The overall exchange quality for a set of instances I_A between schema $A \in Q$ and $B \in Q$ can be computed via $\bar{m}_{AB}$, in which $n_{type(i)}$ is the number of instances of a specific type $i \in I_A$, which is represented as follows:

$$Quality := f(\bar{m}_{AB}, I_A) := \sum_{i \in P(I_A)} \sum_{j \in P(B)} \frac{\bar{m}_{AB}(type(i), j)}{n_{type(i)}} \tag{6.12}$$

Application to the scenario example

The unevaluated schema mapping $m_{c',s}$ of Figure 6.6 has to be extended to an evaluated schema mapping $\bar{m}_{c',s}$ as shown in Figure 6.8. Therefore, each data structure mapping has to be evaluated. This can only be achieved on their attributes as exemplified in Figure 6.7. The attributes l, w of data structure $c_1 : myWall3D$ can be mapped directly to the corresponding attributes l, w of data structure $s_1 : Walls2D$ without any loss of information. The attribute h of data structure $c_1 : myWall3D$ can only be mapped to $\varnothing$ and leads to a loss of information.

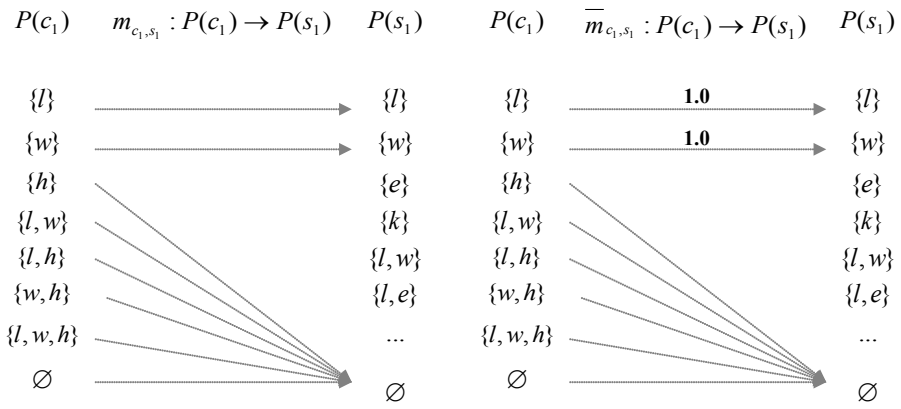


Figure 6.7: Attribute and evaluated attribute mapping.

The quality of a mapping of data structure $c_1 : myWall3D$ to $s_1 : Wall2D$ is:

$$Q_{c_1,s_1} = \frac{\bar{m}(\{l_{c_1}\},\{l_{s_1}\}) + \bar{m}(\{w_{c_1}\},\{w_{s_1}\}) + \bar{m}(\{h_{c_1}\},\varnothing_{s_1})}{1+1+1} = \frac{1.0+1.0+0.0}{3} \approx 0.7$$

The evaluated schema mappings $\bar{m}_{c',s}$ and $\bar{m}_{s,c'}$ to assess bidirectional data exchange are shown in Figure 6.8. An exchange from schema c' to schema s is lossy. The exchange from schema s to schema c' is lossless.

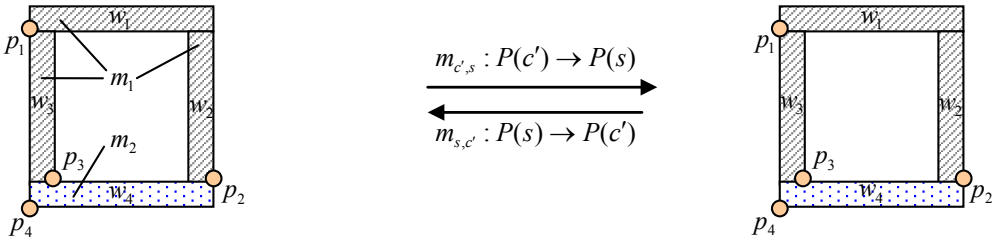
$\overline{m}_{c',s}$	$\{s_1\}$	$\{s_2\}$	$\{s_1,s_2\}$	$\varnothing$
$\{c_1\}$	0.7	0	0	0
$\{c_2\}$	0	0	0	1
$\{c_3\}$	0	0.7	0	0
$\{c_1,c_2\}$	1.0	0	0	0
$\{c_1,c_3\}$	0	0	1.0	0
$\{c_2,c_3\}$	0	0	0	1
$\{c_1,c_2,c_3\}$	0	0	1.0	0
$\varnothing$	0	0	0	1

$\overline{m}_{s,c'}$	$\{c_1\}$	$\{c_2\}$	$\{c_3\}$	$\{c_1,c_2\}$	$\{c_1,c_3\}$	$\{c_2,c_3\}$	$\{c_1,c_2,c_3\}$	$\varnothing$
$\{s_1\}$	0	0	0	1.0	0	0	0	0
$\{s_2\}$	0	0	1.0	0	0	0	0	0
$\{s_1,s_2\}$	0	0	0	0	0	0	1.0	0
$\varnothing$	0	0	0	0	0	0	0	1

Figure 6.8: The evaluated schema mappings $\bar{m}_{c',s}$ and $\bar{m}_{s,c'}$ as matrices.

Finally, the overall quality for specific instance sets can be computed a priori. This is exemplified by instance set $I_{c'}$, which consists of four wall instances w_{1-4} interrelated with two material instances m_{1-2} and four location point instances p_{1-4}.

$I_{c'} = \{w_1, w_2, w_3, w_4, p_1, p_2, p_3, p_4, m_1, m_2\}$ $\qquad$ $I_s = \{w_1, w_2, w_3, w_4, p_1, p_2, p_3, p_4\}$



The overall quality Q can be computed as follows:

$$Q_{c',s} = \frac{4\cdot\bar{m}(\{c_1\},\{s_1\}) + 4\cdot\bar{m}(\{c_3\},\{s_2\}) + 2\cdot\bar{m}(\{c_2\},\varnothing) + 4\cdot\bar{m}(\{c_1,c_2,c_3\},\{s_1,s_2\})}{4+4+2+4}$$

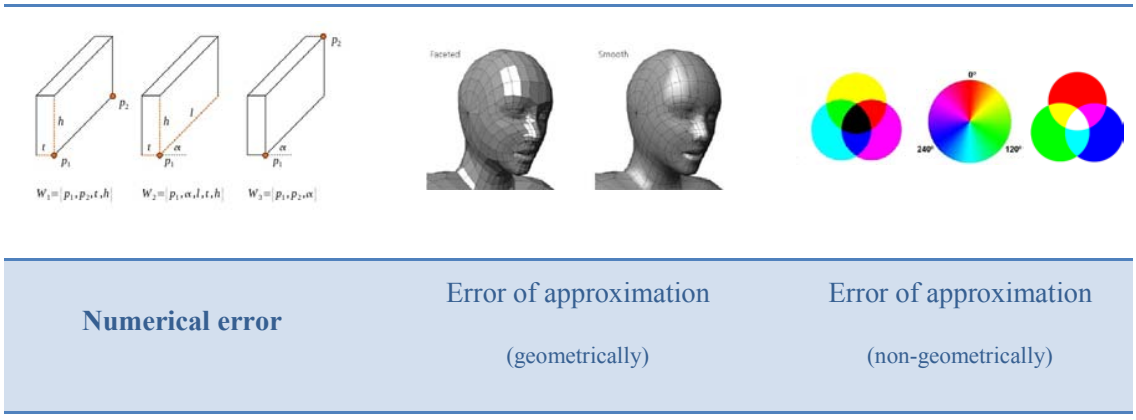
$$Q_{c',s} = \frac{4\cdot 0.7 + 4\cdot 0.7 + 2\cdot 0.0 + 4\cdot 1.0}{4+4+2+4} \approx 0.6857$$

6.4 Uncertainties

A reconsideration of the scenario example shows that the assessment works well for these specific simple scenario conditions. Equivalent model representations and data types with the same precision are also used. The only differences are the reduced geometrical dimension and the variety in the modeling of classes and their associations. In reality, the conditions are more complex and numerous error sources exist. Those errors more or less influence the quality of the mapping and consequently the quality of the data exchange.

6.4.1 Sources of Mapping Errors

Various error sources exist, which lower the quality of the mapping under certain conditions. A widespread error class is the numerical error and the error of approximation. A subclass of the numerical error is the rounding error, e.g., the conversion of floating point values to an integer. Another error subclass arises through the conversion of mathematically exact representations, e.g., of different wall geometry definitions as shown in Figure 6.9 (left). A subclass of an error of approximation is the conversion of more or less conceptually similar data structures, but of a different representation. For example, surfaces can be described via faceted models or by free-form models as shown in Figure 6.9 (middle). The approximation error depends on the mesh quality of the faceted model, however, an error term cannot be avoided. A non-geometrical approximation error is the conversion of different color representations as shown in Figure 6.9 (right). A color can be represented by different color models, such as RGB values, CMY/CMYK values or a web-safe color palette. The conversion of RGB values to the web-safe color palette can lead to an approximation error.



Numerical error	Error of approximation (geometrically)	Error of approximation (non-geometrically)

Figure 6.9: Examples of geometrical and non geometrical errors.

How do file comparison and evaluated schema mapping consider sources of error? File comparison operates on data. Errors can be accurately described by single quality values. In contrast to this, evaluated schema mapping operates on data structures. Within the mapping process, error classes are also detectable, however they can no longer be described by single values. Instead of single values, intervals have to be used. The lower bound describes the worst case of a certain mapping, whereas the upper bound describes the best case.

6.4.2 Interval Arithmetic

Interval arithmetic is used to insert bounds of errors into mathematical computations. It is suitable for a variety of purposes, e.g., to handle rounding errors in calculations or to consider uncertainties with respect to the exact values of physical and technical parameters. Classical arithmetic defines operations on numbers, whereas interval arithmetic defines operations on an interval x, which is expressed as follows:

$$[x_{min}, x_{max}] = \{x \in \mathbb{R} \mid x_{min} \leq x \leq x_{max}\} \tag{6.13}$$

Arithmetic operation $\otimes$ on two intervals $x = [x_{min}, x_{max}]$ and $y = [y_{min}, y_{max}]$ is defined as:

$$\begin{aligned} x \otimes y &= [x_{min}, x_{max}] \otimes [y_{min}, y_{max}] = [min(P), max(P)] \\ P &= \{x_{min} \otimes y_{min}, x_{min} \otimes y_{max}, x_{max} \otimes y_{min}, x_{max} \otimes y_{max}\} \end{aligned} \tag{6.14}$$

For a practical use of Eg. (6.14), the four basic arithmetic operations can be simplified:

- $x + y = [x_{min} + y_{min}, x_{max} + y_{max}]$
- $x - y = [x_{min} - y_{max}, x_{max} - y_{min}]$
- $x \cdot y = [min(P), max(P)],\ P = \{x_{min} \cdot y_{min}, x_{min} \cdot y_{max}, x_{max} \cdot y_{min}, x_{max} \cdot y_{max}\}$
- $x/y = [min(P), max(P)],\ P = \{x_{min}/y_{min}, x_{min}/y_{max}, x_{max}/y_{min}, x_{max}/y_{max}\}$

Finally, a single value r can be also defined as an interval as follows:

$$r \in \mathbb{R} \rightarrow [r, r] \tag{6.15}$$

In general, interval arithmetic enables the development of numerical methods, which yield reliable results. The application to the evaluated schema mapping approach would allow taking into account different sources of errors. Therefore, the use of interval arithmetic in the assessment process is essential.

6.4.3 Adaptation

To apply interval arithmetic, a redefinition of R – from single to interval values – is necessary, Eg. (6.16). Each mapping in $\bar{m}$ is now associated with the interval quality $r \in R$. The bounds can be estimated, e.g., on basis of the experience or knowledge of an engineer or by error estimation (subsection 6.4.4). In addition, the overall quality is also an interval. It contains the worst (x_{min}) and best case (x_{max}) for an expected data transfer of a given set of instances.

$$R := \{(x_{min}, x_{max})] \in \mathbb{R} \times \mathbb{R} \mid 0 < x_{min} \leq 1 \wedge 0 < x_{max} \leq 1 \wedge x_{min} \leq x_{max}\} \quad (6.16)$$

User-Defined Quality Indicators

Moreover, schema mapping can be classified by the following two additional parameters ϵ and ϕ. The parameter ϵ describes the variation in quality, which the user still finds acceptable. It allows a classification of overall quality, according to the range $[x_{min}, x_{max}]$:

- If $x_{min} = x_{max}$, then the mapping is well defined (no uncertainties are included).
- If $(x_{max} - x_{min}) \leq \epsilon$, then the mapping is robust (the uncertainties are within the acceptable variation in quality values).
- If $(x_{max} - x_{min}) > \epsilon$, then the mapping is sensitive (the uncertainties are outside of the acceptable variation in quality values).

The parameter ϕ describes the quality of the data exchange, which the user still finds acceptable. It allows a categorization of the mapping according to the quality values:

- If $x_{max} = 0$, then the quality of the mapping is the worst case. The user is advised to redesign the model or to change the software application.
- If $x_{max} < \phi$, then the quality of the mapping is not acceptable to the user. The user is advised to redesign the model or to change the software application.
- If $x_{min} < \phi \wedge x_{max} > \phi$, then the quality of the mapping may be acceptable to the user. The user is advised to use an a posteriori approach to compute the real quality.
- If $x_{min} \geq \phi$, then the quality of the mapping is acceptable to the user.
- If $x_{min} = 1$, then the quality of the mapping is the best case.

6.4.4 Example

Problem statement: The mapping quality for prismatic and cylindric columns should be estimated. Source system A uses a facet modeler and a free-form modeler to describe the columns as accurately as possible. The modeler of system B can only handle faceted solids. An exchange quality can be estimated based on the differences in their volumes (Figure 6.10).

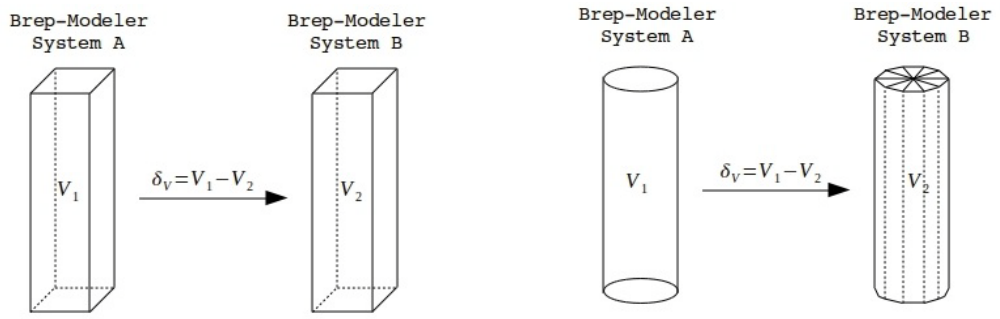


Figure 6.10: Error estimation based on volume.

- The difference δ_V between the volume V_1 of a column in the source system and the volume V_2 of the same column in the target system can by computed as:

$$\delta_V = |V_1 - V_2|$$

- The absolute error ϵ can by computed as the ratio of δ_V to V_1:

$$\epsilon = \frac{\delta_V}{V_1}$$

- Finally, the evaluated mapping quality $\bar{m}_{A_{col}B_{col}}$ can be computed as:

$$\bar{m}_{A_{col}B_{col}} = 1 - \epsilon$$

Application:

- In the case of an exchange of rectangular columns there is no difference in the volumes $\delta_V = 0$ as shown in Figure 6.10 (left). This results in $\epsilon = 0$ since both modelers can handle faceted solids without a change in the representation of the column. The evaluated mapping quality of rectangular columns from system A to system B is:

$$\bar{m}_{A_{rect}B_{rect}} = [1.0, 1.0]$$

- In the case of an exchange of cylindrical columns there is a difference in the volumes $\delta_V \neq 0$ as shown in Figure 6.10 (right). The reason is that the modeler of the target system cannot handle free-form solids; instead, it has to approximate the cylinder as an n-prism. The mapping quality depends on the base n of the prism:

$$\delta_V = \left| \pi \cdot r^2 \cdot h - \frac{n \cdot r^2}{2} \cdot sin\frac{360}{n} \cdot h \right| = \left| \pi - \frac{n}{2} \cdot sin\frac{360}{n} \right|, \; \epsilon = 1 - \left| \frac{\frac{n}{2} \cdot sin\frac{360}{n}}{\pi} \right|$$

This results in the following mapping quality graph as shown below:

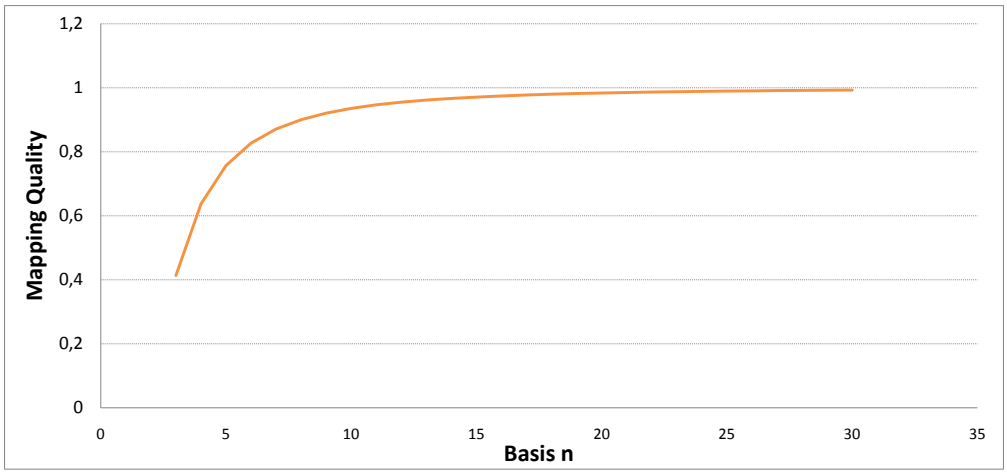


Figure 6.11: Mapping quality of cylindrical columns as a function of n.

Example 1: The approximation of cylindrical columns in system A through n-prism columns of system B – without any schema restrictions – would result in $(n = 3)$ to $\epsilon = 0.587$ in the worst case, and in $(n = \infty)$ to $\epsilon = 0.0$ in the best case. The evaluated mapping of circular columns from system A to system B without restrictions is:

$$\bar{m}_{A_{circ}B_{prism}} = [0.413, 1.0]$$

Example 2: In practice, the approximation of cylindrical columns in system A through n-prism columns of system B with a base of $n = 3$ is not common, and with a base of $n = \infty$ it is not realizable. Usually, schema restrictions exist for defining the lower and upper limits of base n. The evaluated mapping of circular columns from system A to system B, e.g., for $n = 20$ (lower limit) and $n = 100$ (upper limit) is:

$$\bar{m}_{A_{circ}B_{prism}} = [0.984, 0.99]$$

7 Adaptation to the OO Paradigm

One widespread modeling paradigm in computer science is the object-oriented paradigm. In the context of the schema mapping approach (section 6.3), classes are related to data structures and objects are related to particular instances of classes, which contain the data of a certain state. The schema S and the instance set I_A can now be written as follows:

$$S := \{e | e \text{ is a class}\} \in Q$$

$$I_A := \{i \mid i \text{ is an object of a class } e \in A \in Q\}$$

7.1 Problem Statement

Classes inside a schema can be interrelate to describe more complex facts. These relationships are included in the power set. The power set $P(S)$ of any set S is the set of all subsets of S, including the empty set and S itself. Mathematically, the power set contains $|P(S)| = 2^n$ subsets, in which n is the number of elements in S. The number of subsets is doubled when the number of elements increases by 1 (Figure 7.1).

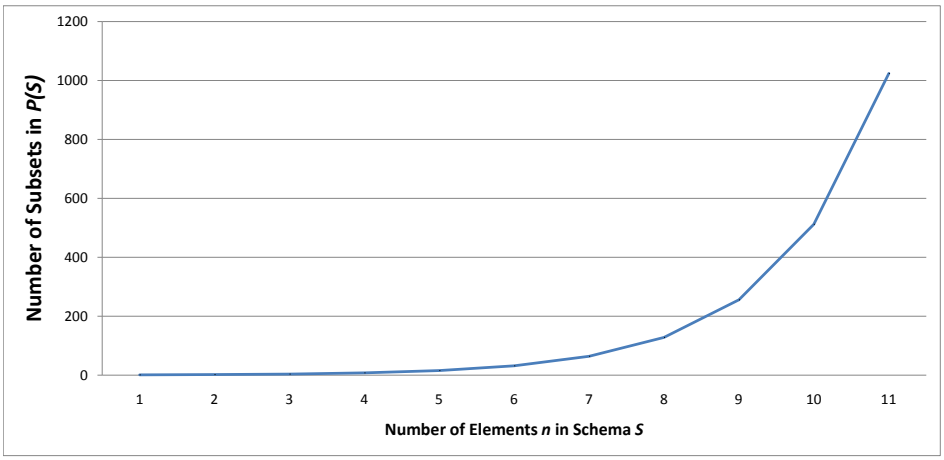


Figure 7.1: Number of subsets of $P(S)$ depending on n.

Unfortunately, data schemas generally contain a multiplicity of classes. In the construction industry, the number of elements easily grow to hundreds of classes. The reasons for this are the complexity of the tasks (architecture, engineering, construction, facilities management), the multiple phases of the construction project life cycle, and the involvement of multidisciplinary teams and perspectives (owners, architects, consultants, engineers, etc.). The problem statement is clarified in the following example:

IFC Schema: *The IFC data schema intends to describe building and construction data. It is widely used for Building Information Modeling (BIM). The IFC is defined in EXPRESS and structured in a deep object-oriented inheritance model consisting of 759 entities (Vers.2x4). From a mathematical point of view, the power set of the current IFC schema results in an unmanageable number of entities, which is expressed as follows:*

$$|P(IFC_{2x4})| = 2^{759} = 3.03 \cdot 10^{228} \tag{7.1}$$

7.2 Solution Approach

From a mathematical point of view, the power set contains all the possible class relations with respect to the single classes of the schema. However, from a software engineering point of view, the classes of a schema S are only in a relationship with the subset of classes $T \subseteq P(S)$. This is exemplified by the following simple scenario:

An object-oriented software application needs three classes A, B and C (Figure 7.2) to describe their data. This results in the following data schema: $S = \{A, B, C\}$.

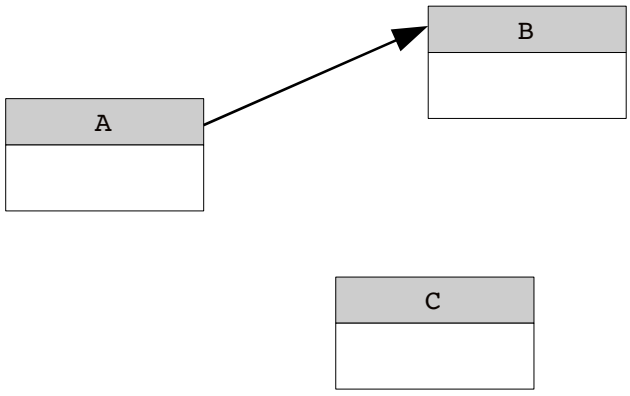


Figure 7.2: Class diagram of schema S in UML notation.

- *From a mathematical point of view, the power set is comprised of the empty set $\varnothing$, the schema S itself and all the combinations of class relations. The maximum number of entities is $|P(S)| = 2^3 = 8$:*

$$P(S) = \{\varnothing, \{A\}, \{B\}, \{C\}, \{A, B\}, \{A, C\}, \{B, C\}, \{A, B, C\}\}$$

- *From a software engineering point of view, an analysis of schema S regarding the existing class relations shows that a class relation between classes A and C ($\{A, C\}$), between classes B and C ($\{B, C\}$) and between classes A, B and C ($\{A, B, C\}$) does not exist. All non-existing class relations of schema S are collected in the set $\bar{P}(S)$:*

$$\bar{P}(S) = \{\{A, C\}, \{B, C\}, \{A, B, C\}\} \subseteq P(S)$$

- *Finally, the set difference of power set $P(S)$ and $\bar{P}(S)$ results in a new modified set $P(S)_{real}$, which contains only the real existing class relations of schema S:*

$$P(S)_{real} = P(S) \setminus \bar{P}(S) = \{\varnothing, \{A\}, \{B\}, \{C\}, \{A, B\}\}$$

The removal of non-existing class relations becomes an essential aspect since high numbers of class relations would result from large schemas, e.g., the IFC schema. The identification of $\bar{P}(S)$ via a schema analysis by taking into account the basic principles of the object-oriented paradigm is discussed in the next section.

7.3 Schema Analysis

A relationship is a general term covering the specific types of logical connections found in a class. The object-oriented paradigm provides different mechanisms for generating relationships between classes, such as general relations, instance and class-level relations, and abstract data types. Each type has more or less influence on $\bar{P}(S)$. The relationships between classes are encoded in the schema itself. Decoding the relationships involved via a schema analysis and an examination of the degree of influence are part of the next subsections. In each subsection, a scenario is used to describe and analyze the relationship to be considered. It starts with the simplest scenario, increases in complexity stepwise by adding other types of relationships.

7.3.1 General Relationship

A general relationship is the weakest form of relationship in the object-oriented paradigm. It indicates that one class depends on another. Dependency exits if a class is the parameter or local variable of a method that belongs another class. In UML, a general relationship between two classes A and B is represented by a dashed line, the optional label *«use»* and an arrowhead pointing towards the class used, as shown in Figure 7.3. The general relationship in Figure 7.3 can be translated as: *class A uses class B at some point of time*.

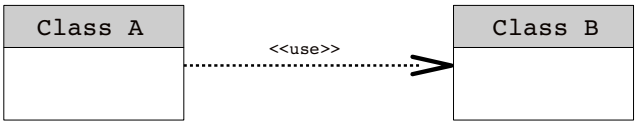


Figure 7.3: General relationship between two classes.

Scenario: An object-oriented software application in the field of CAD has to deal with different building elements (e.g., wall and column), which are linked with additional material properties. Furthermore, a location point is needed to position the building elements. One possible schema is shown in Figure 7.4:

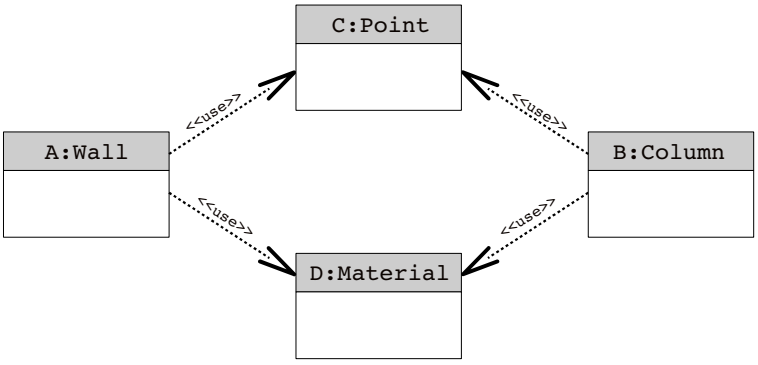


Figure 7.4: Class diagram of schema S.

Analysis: Building elements, material properties and the location point are described as single pieces of information using single classes (Figure 7.4). This leads to a schema S consisting of the following four classes: ***A***:$Wall$, ***B***:$Column$, ***C***:$Point$ and ***D***:$Material$.

$$S = \{A, B, C, D\}$$

Complex facts are encoded in the schema itself as multiple combinations of single classes implemented via general relationships.

As an example, the aspect that building elements can be linked with a material is included in the class combinations $\{A, D\}$ *and* $\{B, D\}$*. The aspect that building elements can be linked with a location point is included in* $\{A, C\}$ *and* $\{B, C\}$*. In addition, building elements can also be linked with both a material and location point. This is reflected in* $\{A, C, D\}$ *and* $\{B, C, D\}$*. Mathematically, all these aspects are described in the power set* $P(S)$ *of the schema* S*. The number of entities is* $|P(S)| = 2^4 = 16$*:*

$$P(S) = \{\varnothing, \{A\}, \{B\}, \{C\}, \{D\}, \{A, B\}, \{A, C\}, \{A, D\}, \{B, C\}, \{B, D\}, \{C, D\}, \{A, B, C\}, \{A, B, D\}, \{A, C, D\}, \{B, C, D\}, \{A, B, C, D\}\}$$

However, from a software engineering point of view, the classes of a schema are only related to a subset of classes. An analysis of schema S according to general relationships shows that some entities within the power set cannot be reflected by the schema.

As an example, the classes A *and* B*, as well as the classes* C *and* D*, are not in a relationship with each other. In this case, the entities* $\{A, B\}$ *and* $\{C, D\}$ *will never be activated during the mapping and evaluation process, therefore, they can be removed from the original power set. The complete set* $\bar{P}(S)$ *of removable entities after schema analysis is* $\left|\bar{P}(S)\right| = 5$*:*

$$\bar{P}(S) = \{\{A, B\}, \{C, D\}, \{A, B, C\}, \{A, B, D\}, \{A, B, C, D\}\}$$

Finally, the set difference $P(S) \setminus \bar{P}(S)$ results in the new set $P(S)_{real}$, which contains only the real existing classes and the class relations of schema S:

$$P(S)_{real} = \{\varnothing, \{A\}, \{B\}, \{C\}, \{D\}, \{A, C\}, \{A, D\}, \{B, C\}, \{B, D\}, \{A, C, D\}, \{B, C, D\}\}$$

Conclusion: The scenario shows that the number of entities to be examined can be reduced by analyzing the general relationships between the classes of a schema. In this case, the number of entities of the original power set $P(S)$ is reduced by more than 30 %.

7.3.2 Instance-Level Relationships

Instance-level relationships are relationships among objects. They are usually diagrammed as a line and connected to a class at each end, as shown in Figure 7.5. In addition, they can be labeled as follows: *"has a"*, *"needs a"*, and *"owns a"*. The ends of the line can also be marked with additional properties, such as role names, ownership indicators, multiplicity and visibility. Higher order types are aggregation and composition (Figure 7.6 and 7.7).

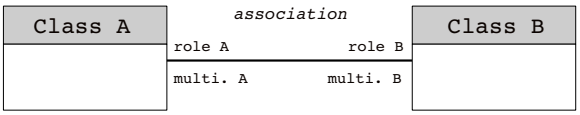


Figure 7.5: Association between two classes.

Aggregation is a *part-whole* or *part-of* relationship. It occurs when a class is a container or a collection of other classes. The dependency between the container class and the contained classes is weak, which means that if the container is destroyed, its contents are not. In this case, the link between the aggregated classes is **optional** and unimportant for the life cycle. It is represented as a hollow diamond shape, which is attached to the class to be contained. The aggregation in Figure 7.6 can be translated as: *class A has a class B*.

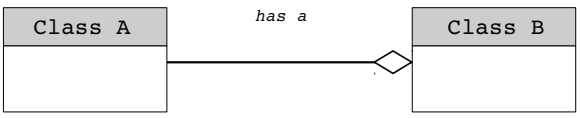


Figure 7.6: Aggregation between two classes.

Composition is more specific than aggregation. The dependencies between the container class and the contained classes are strong, which means that if the container is destroyed, each instance contained in it is destroyed as well. The link between the composed classes is **required** and important for the life cycle. It is represented as a filled diamond shape, which is attached to the class to be contained. The composition in Figure 7.7 can be translated as: *class A owns a class B*.

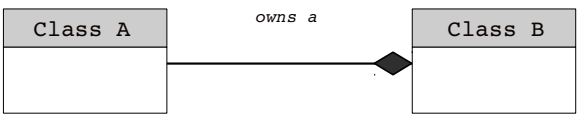


Figure 7.7: Composition between two classes.

Scenario: The scenario from subsection 7.3.1 (Figure 7.4 on page 86) is reused without any changes to the classes or their relationships. The existing general relationships are analyzed according to instance level relationships. The new class design is shown in Figure 7.8 below:

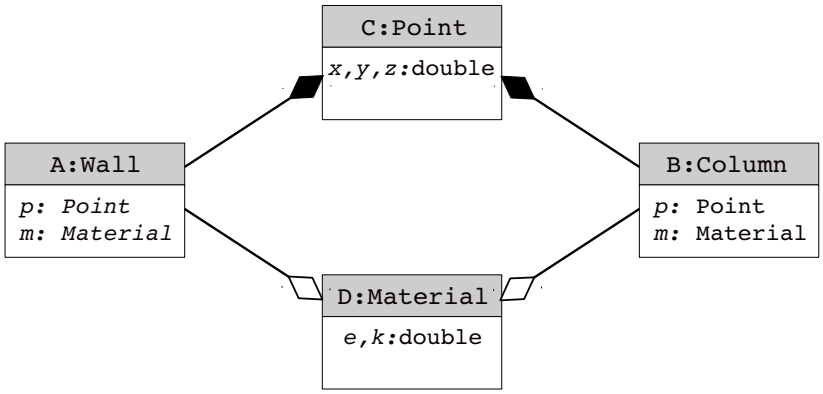


Figure 7.8: Schema S according to instance-level relationships.

- *The material properties within the software system should be able to be removed without any influence on the life cycle of building elements and vice versa. Therefore, the relationship between the building elements and the material is weak and optional; the general relationships between classes A and D, as well as between classes B and D are replaced by an aggregation.*

- *A removal of building elements within the system results directly in the removal of the assigned location point and vice versa. The life cycle is influenced by both. The relationship between the building elements and the location point is strong and required; the existing general relationships between classes A and C, as well as between classes B and C are replaced by a composition.*

Analysis: The scenario is reused without any changes to the classes. Thus, there are no changes on schema $S = \{A, B, C, D\}$ and hence no changes to $P(S)$:

$$P(S) = \{\varnothing, \{A\}, \{B\}, \{C\}, \{D\}, \{A, B\}, \{A, C\}, \{A, D\}, \{B, C\}, \{B, D\}, \\ \{C, D\}, \{A, B, C\}, \{A, B, D\}, \{A, C, D\}, \{B, C, D\}, \{A, B, C, D\}\}$$

Moreover, complex facts are also encoded in the schema itself as a multiple combination of single classes, which are now implemented via aggregation and composition.

- The aspect that a building element can be linked with a material $\{A, D\}, \{B, D\}$ is now described through an optional link via aggregation. Therefore, the life cycle of building elements and materials is independent of both. It follows then that building elements $\{A\}, \{B\}$ and material $\{D\}$ can also be within the software system as single pieces of information.

- In contrast, the aspect that a building element can be linked with a location point $\{A, C\}, \{B, C\}$ is now described through a required link via composition. Therefore, a building element needs a location point as the information required for it to exist and vice versa. The life cycle of building elements and the location point is dependent on both. As a result, the building elements $\{A\}$, $\{B\}$ and the location point $\{C\}$ cannot exist separately, hence they are added to $\bar{P}(S)$.

- Furthermore, all the entities in $P(S)$ that comprise the building element A or B are invalid, however but not the required location point. The invalid elements have to be added to $\bar{P}(S)$.

It should be noted that from a local point of view, an analysis of aggregation and composition may result in conflicts between the two, which is explained in the following example: *an analysis of aggregation leads to the allowed entities:* $\{A\}$, $\{B\}$, $\{D\}$, $\{A, D\}$ *and* $\{B, D\}$. *However, the composition does not allow the entities* $\{A\}$, $\{B\}$, $\{A, D\}$ *and* $\{B, D\}$. *This conflict can be resolved by taking into account the fact that the effect of composition is stronger than the effect of aggregation.* The set $\bar{P}(S)$ after a schema analysis is $\left|\bar{P}(S)\right| = 10$:

$$\bar{P}(S) = \{\{A\}, \{B\}, \{C\}, \{A, B\}, \{A, D\}, \{B, D\}, \{C, D\}, \\ \{A, B, C\}, \{A, B, D\}, \{A, B, C, D\}\}$$

Finally, the set difference $P(S) \setminus \bar{P}(S)$ results in $|P(S)_{real}| = 6$ entities:

$$P(S)_{real} = \{\varnothing, \{D\}, \{A, C\}, \{B, C\}, \{A, C, D\}, \{B, C, D\}\}$$

Conclusion: The use of strong associations between classes has a significant influence on the number of entities in $\bar{P}(S)$. This is caused by the fact that single classes are not permitted and that only specific class combinations are possible. The scenario showed that the number of entities to be examined can be drastically reduced by analyzing the instance-level relationships between the classes of a schema. In this case, the number of entities of the original power set $P(S)$ is reduced by over 60 %.

7.3.3 Class-Level Relationships

Class-level relationships are relationships among classes. One specific type is the generalization known as inheritance, in which one class (child or subclass) is based on another (parent or superclass). It is commonly used to capture properties (attributes, operations and relationships) in a parent class for further reuse in all of the associated child classes. Generalization is implemented via an inheritance of properties from the parent to all of the children. Inheritance refers to the ability of one class (child) to inherit the identical functionality of another class (parent), and then add new functionality to its own. The child defines only the properties that are distinct from the parent. In the UML, generalization relationships do not have a name. They are diagrammed as a solid line with a hollow arrowhead that points from the child class to the parent class. In Figure 7.9, the generalization relationship between the parent class A and the associated child classes B and C, is shown. The aggregation defined between class A and class D will be inherited to all the child classes B and C. Thus, the classes B and C are also linked with class D by an aggregation. The generalization can be translated as: *class **B** is a specific class **A*** and *class **C** is a specific class **A***.

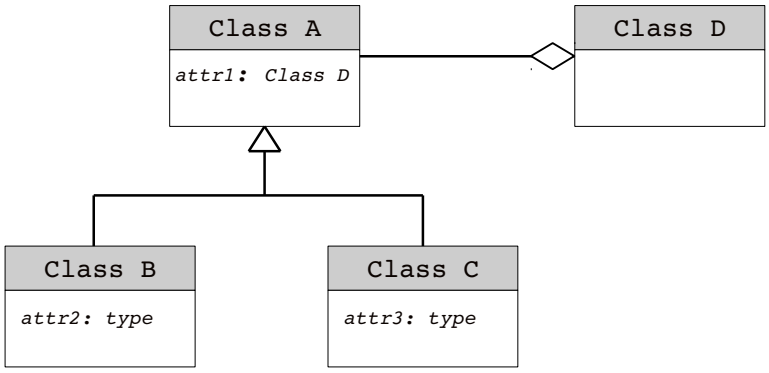


Figure 7.9: Generalization between one superclass A and two subclasses B, C.

Scenario: From a software engineering point of view, the scenario from subsection 7.3.2 (Figure 7.8 on page 89) is modified a little in order to examine the influence of class-level relationships on $\bar{P}(S)$.

- Class A (which represents a *Wall*) and class B (which represents a *Column*) are semantically similar. Both classes can be classified as a specific building element. From a software engineering point of view, the use of generalization within the scenario is recommended. Therefore, a new class E (Element) is inserted into schema S in order to describe a general building element.

- Classes A and B are linked to class E via a class-level relationship (generalization). For now, E acts as the parent for all specific building elements within the software system. Therefore, the properties that all children have in common can be moved to the parent.

- An examination of the existing child properties leads to the result that A and B are linked to D via aggregation and to C via composition. Thus, these relationships are in common and can be moved from the child classes A and B to the parent class E. The information that a wall and a column is linked to a material and to a location point is now included in the parent class E, but it is also inherited by all children.

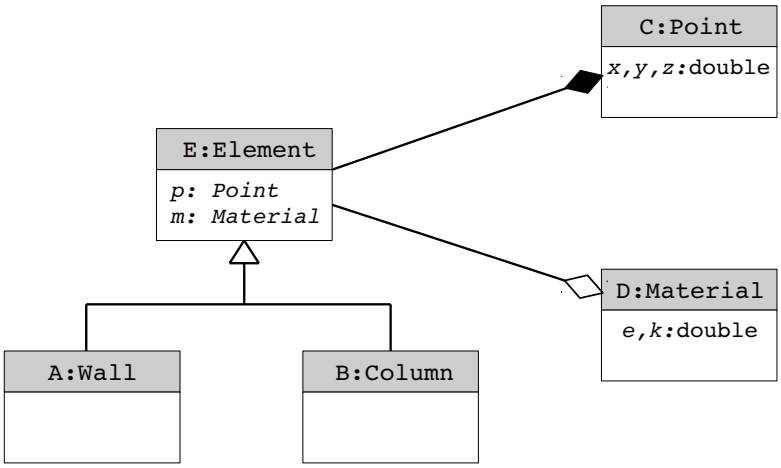


Figure 7.10: Class diagram of schema S in UML notation.

Analysis: The scenario was modified a little by inserting a new class E. Thus, the schema S now contains 5 entities:

$$S = \{A, B, C, D, E\}$$

However, the number of combinations in $P(S)$ is directly influenced by the number of classes in schema S. In this case, the number is doubled $|P(S)| = 2^5 = 32$:

$$\begin{aligned} P(S) = \{&\varnothing, \{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{A, B\}, \{A, C\}, \{A, D\}, \{A, E\}, \{B, C\}, \\ &\{B, D\}, \{B, E\}, \{C, D\}, \{C, E\}, \{D, E\}, \{A, B, C\}, \{A, B, D\}, \{A, B, E\}, \\ &\{A, C, D\}, \{A, C, E\}, \{A, D, E\}, \{B, C, D\}, \{B, C, E\}, \{C, D, E\}, \\ &\{B, D, E\}, \{A, B, C, D\}, \{A, B, C, E\}, \{A, B, D, E\}, \\ &\{A, C, D, E\}, \{B, C, D, E\}, \{A, B, C, D, E\}\} \end{aligned}$$

The complex aspects now are described via instance-level relationships (aggregation and composition) and class-level relationships (generalization).

- The aspect that a building element E can be linked with a location point C is described by composition. Therefore, the entities $\{E\}$ and $\{C\}$ cannot exist separately. They are added to $\bar{P}(S)$. In addition, E is the parent class of A and B and thus they inherit these compositions as well. Consequently, the entities $\{A\}$ and $\{B\}$ also cannot exist separately and have been added to $\bar{P}(S)$.

- Furthermore, all the entities in $P(S)$, which comprise A, B or E, but not the required location point C, are also added to $\bar{P}(S)$.

- Generalization relationships are also reflected within $P(S)$. The generalization of a wall A and a column B with respect to the general building element E are represented by the combinations $\{A, E\}$ and $\{B, E\}$. However, from a software engineering point of view, these are not permitted. Instances of A, B and E can only be created as single objects. Therefore, all the entities, which comprise such generalization relationships have to be added to $\bar{P}(S)$ as well.

The complete set $\bar{P}(S)$ of entities that can be removed after schema analysis is $\left|\bar{P}(S)\right| = 24$:

$$\begin{aligned}\bar{P}(S) = \{&\{A\}, \{B\}, \{C\}, \{E\}, \{A, B\}, \{A, D\}, \{A, E\}, \{B, D\}, \{B, E\}, \{C, D\}, \{D, E\},\\ &\{A, B, C\}, \{A, B, D\}, \{A, B, E\}, \{A, C, E\}, \{A, D, E\}, \{B, C, E\},\\ &\{B, D, E\}, \{A, B, C, D\}, \{A, B, C, E\}, \{A, B, D, E\},\\ &\{A, C, D, E\}, \{B, C, D, E\}, \{A, B, C, D, E\}\}\end{aligned}$$

Finally, the set difference $P(S) \setminus \bar{P}(S)$ results in $|P(S)_{real}| = 8$ entities:

$$P(S)_{real} = \{\varnothing, \{D\}, \{E, C\}, \{A, C\}, \{B, C\}, \{E, C, D\}, \{A, C, D\}, \{B, C, D\}\}$$

Conclusion: The use of generalization within the scenario primarily increases the number of classes in S and consequently also increases the number of combinations in $P(S)$. Fortunately, generalization relationships and all the combinations, which comprise generalization relationships can be removed from $P(S)$. Therefore, in this scenario the number of entities to be examined can be reduced by more than 75 %.

7.3.4 Abstract Data Types

Abstract data types[1] are purely theoretical entities. They are used to classify data structures and/or to simplify the implementation of abstract algorithms. The creation of instances of an abstract data type is not permitted, therefore it is not supported by most object-oriented programming languages. The content and behavior of an abstract class within a software system can only be used by a non-abstract class, which is based on the abstract class. This can be achieved through generalization relationships. Consequently, generalization relationships and abstract classes are strongly interrelated in the object-oriented paradigm. The parent class is the most appropriate for being used as an abstract class. In the UML, an abstract class is flagged by the additional label *"abstract"* below the class name as shown in Figure 7.11.

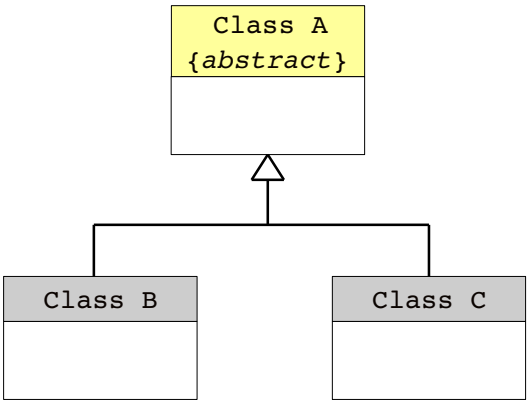


Figure 7.11: Composition between two classes.

Scenario: From a software engineering point of view, the scenario from subsection 7.3.3 (Figure 7.10 on page 92) is modified according to abstract classes.

- Abstract classes are theoretical entities, which are mostly used for conceptual reasons. They are strongly interrelated with the aspect of generalization. For this reason, their use requires the existence of generalization relationships between classes. In addition, the parent class is the most appropriate class for being flagged as abstract. An analysis of the scenario leads to the fact that only class E fulfills all the requirements. Therefore, the general parent class E is flagged as an abstract class (Figure 7.12).

[1] Realized as abstract classes in the object-oriented paradigm.

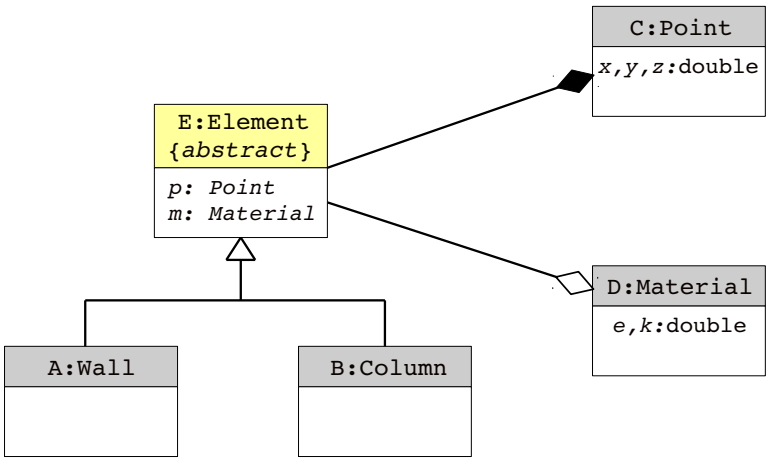


Figure 7.12: Class diagram of schema S in UML notation.

Analysis: The scenario is marginally modified by changing class E to an abstract class. Therefore, the classes in schema S, as well as the entities of $P(S)$, are identical to the scenario in subsection 7.3.3. Complex aspects are now described via instance-level and class-level relationships, as well as by abstract classes.

- The result $\bar{P}(S)$ of removable combinations with respect to instance-level and class-level relationships is identical to the result presented in subsection 7.3.3.

- Additionally, abstract classes cannot be utilized as a template for objects, therefore they are not available as a piece of information within the software system. Therefore, each combination in $P(S)$, which comprises the abstract class E has to be added to $\bar{P}(S)$.

The complete set $\bar{P}(S)$ of entities that can be removed after a schema analysis is $\left|\bar{P}(S)\right| = 26$:

$$\begin{aligned}\bar{P}(S) = \{&\{A\},\{B\},\{C\},\{E\},\{A,B\},\{A,D\},\{A,E\},\{B,D\},\{B,E\},\{C,D\},\{D,E\},\\ &\{E,C\},\{A,B,C\},\{A,B,D\},\{A,B,E\},\{A,C,E\},\{A,D,E\},\{B,C,E\},\\ &\{B,D,E\},\{E,C,D\},\{A,B,C,D\},\{A,B,C,E\},\{A,B,D,E\},\\ &\{A,C,D,E\},\{B,C,D,E\},\{A,B,C,D,E\}\}\end{aligned}$$

Finally, the set difference $P(S) \setminus \bar{P}(S)$ results in $|P(S)_{real}| = 6$ entities and is identical to the result in subsection 7.3.2:

$$P(S)_{real} = \{\varnothing,\{D\},\{A,C\},\{B,C\},\{A,C,D\},\{B,C,D\}\}$$

Conclusion: Abstract classes cannot be utilized as pieces of information and can also be removed from schema S before starting a schema analysis. As a result, the entities to be examined, which are collected in $P(S)_{real}$, are identical to the entities of subsection 7.3.2. For every abstract class within S, the number of entities in $P(S)$ is halved.

Please note: Abstract classes can also be used with respect to instance-level relationships as exemplified in Figure 7.13. Therefore, the correlation between associations and abstract classes must be taken into account. As an example, a class F has been introduced in order to describe a building consisting of several building elements E. Consequently, the classes F and E are interrelated by association. This is reflected by the combinations $\{F, E\} \in P(S)$. However, class E is flagged as abstract and can be removed from schema S, however combination $\{F, E\}$ first has to be resolved. This resolving can only occur on the basis of its children. The parent E has to be replaced by its children, A and B. This is reflected in $P(S)$ by the combinations $\{F, A\}$ and $\{F, B\}$.

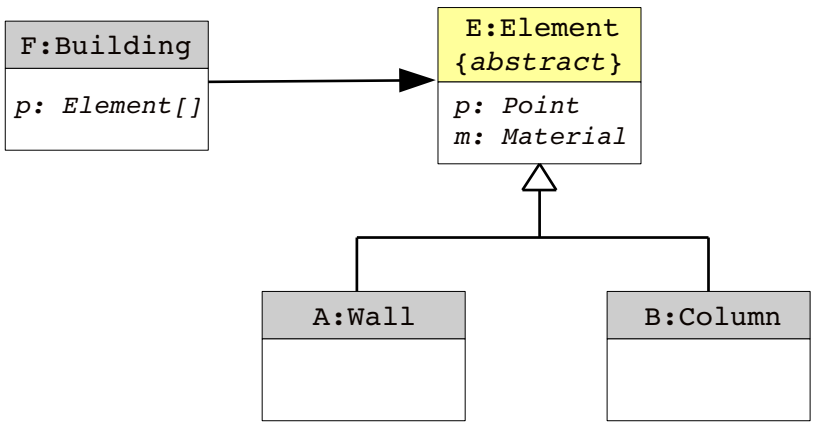


Figure 7.13: Association of an abstract class.

7.4 Conclusion

Classes can be interrelated to describe more complex facts. These relationships are included in the power set $P(S)$. Mathematically, the power set contains 2^n subsets, in which n is the number of elements in S. Unfortunately, data schemas, in general, contain a multiplicity of classes and thus the power set results in an unmanageable number of entities. From an software engineering point of view, the classes of a schema are only in relation to a subset of classes. Therefore, performing a schema analysis is an essential factor in the proposed

schema mapping approach. It can be shown that the number of entities to be examined in the mapping and evaluation process can be drastically reduced. The influence of the different types of relationships is summarized below:

Influence on

Type of Relationship	$P(S)$	$\bar{P}(S)$	$P(S)_{real}$
General			
Dependency	No	Low	Low
Instance-level			
Aggregation (optional)	No	Low	Low
Composition (required)	No	High	Low-High
Class-level			
Generalization	Increased	High	High
Abstract Classes	Decreased	Low	High

Figure 7.14: The influence of relationships with respect to schema analysis.

- From a local point of view, a schema analysis can be conducted stepwise according to the different types of relationships. Each analysis results in its own set $\bar{P}_i(S)$ of removable entities and consequently in a different $P_i(S)_{real}$ set.
- From a global point of view, the set $P(S)_{real}$ can be computed as the difference of the original $P(S)$ and the sum of all sets $\bar{P}_i(S)$ of removable entities:

$$P(S)_{real} = P(S) \setminus \left(\bar{P}_1(S) \cap \bar{P}_2(S) \cap ... \cap \bar{P}_i(S)\right) = P(S) \setminus \left(\bigcap_i \bar{P}_i(S)\right) \quad (7.2)$$

8 Quality Assessment

The proposed approach for assessing data exchange has been applied to the multi-story frame structure shown in Figure 8.1. It is one of the three reference objects dealt with in the research group. The frame structure is suitable for examining essential problems, such as bracing systems, soil-structure interaction and the impact of earthquakes [3]. Therefore, it has been adopted within various of the subprojects of the research group. Because of the different tasks, the frame structure has to be modeled at various levels of detail, regarding geometry, material, masonry infill, kinematics, soil-structure interaction, etc. The elements of the frame structure have been classified into primary and secondary elements.

- Primary elements are basic elements, like beams and columns. They are an inherent part of the modeling of the frame structure.
- Secondary elements are additional elements, like braces, panels, foundation, soil, bearing, loads, and material. They play an optional part in modeling the frame structure. Their use is dependent on the task to be examined.

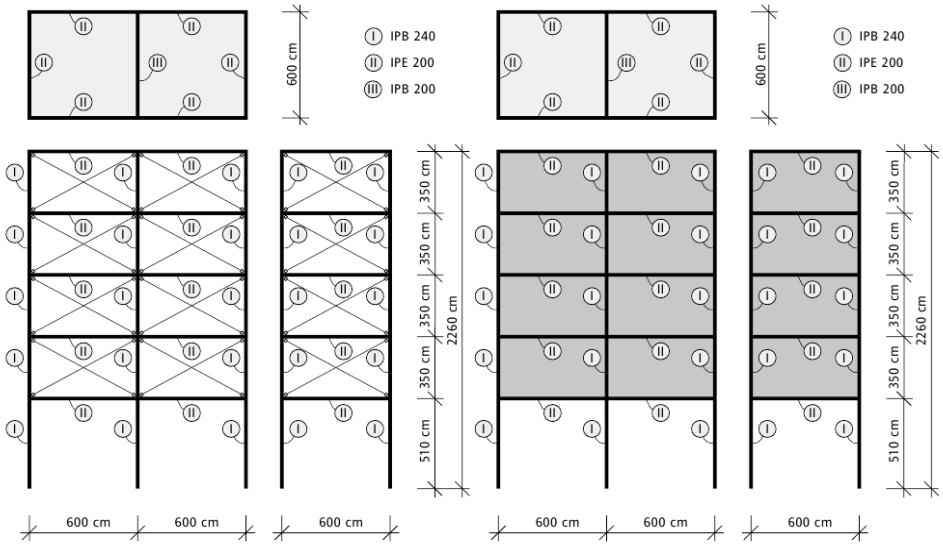


Figure 8.1: The multi-story frame structure with braces (left) and panels (right).

8.1 Coupling Scenario

Within the research group, the frame structure has been adopted in order to investigate certain sub-problems with the aid of various software applications.

- ANSYS, which is a FEM-based engineering simulation software, has been used for developing an evaluation method for the prognosis quality of complex engineering models. In addition, it has been used for static linear calculations in the context of adaptive structural design considering soil-structure interaction.

- SAP2000, which is another FEM-based engineering simulation software, has been used in order to perform dynamic non-linear calculations in the context of adaptive structural design considering soil-structure interaction.

- Additional software applications with different domains, such as CADEMIA (engineering application), SLANG (stochastic modeling), PLAXIS (geotechnical analysis) or SYSWELD (welding simulation), which have already been used within the research group, may also be linked to the coupling scenario.

Due to the various software applications involved, a centralized approach is suggested in order to enable collaboration and systems integration. In addition, the Industry Foundation Classes that have been developed to describe building and construction industry data were chosen for the central building data model, which results in the following coupling scenario:

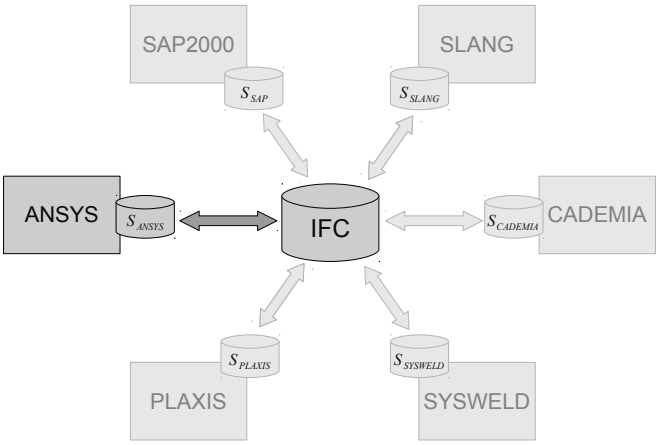


Figure 8.2: Centralized coupling scenario.

The suggested coupling scenario consists of the following seven participating schemas:

$$Q = \{IFC, S_{ANSYS}, S_{SAP}, S_{SLANG}, S_{CADEMIA}, S_{SYSWELD}, S_{PLAXIS}\}$$

and the following twelve schema couplings, i.e. twelve schema analyses and mappings have to be made:

$$\begin{aligned} C = \{&(IFC, S_{ANSYS}), (S_{ANSYS}, IFC), (IFC, S_{SAP}), (S_{SAP}, IFC), (IFC, S_{SLANG}), \\ &(S_{SLANG}, IFC), (IFC, S_{CADEMIA}), (S_{CADEMIA}, IFC), (IFC, S_{SYSWELD}), \\ &(S_{SYSWELD}, IFC), (IFC, S_{PLAXIS}), (S_{PLAXIS}, IFC)\} \end{aligned}$$

Schema analysis and schema mapping within each schema coupling are important for assessing the quality of data exchange and meeting the requirements for collaboration and systems integration.

The proposed approach for assessing the quality of data exchange has been applied for schema coupling, as exemplified by $(IFC, S_{ANSYS}) \in C$.

8.2 Schema Coupling: $(\mathbf{IFC}, \mathbf{S_{ANSYS}}) \in \mathbf{C}$

Schema coupling $(IFC, S_{ANSYS}) \in C$ includes two different schemas, the IFC schema and the ANSYS schema. Linking both schemas successfully would allow ANSYS to access commonly shared building data, for example, in order to enable linear calculations for adaptive structural design considering soil structure interaction.

However, mapping both schemas is a challenge since they follow different paradigms in defining data structures. The IFC schema is a set of neutral object-oriented entities structured in a deep inheritance graph, whereas the ANSYS schema is a set of commands forming a proprietary scripting/macro language. The schema mapping will be achieved through a mapping between IFC entity data types and ANSYS macros/commands.

The different data types and logical connections, which were defined by the schemas involved have to be known in order to carry out the schema analyses. For this reason, the IFC and ANSYS schema are briefly introduced.

8.2.1 IFC Data Schema

The IFC data schema is a set of neutral, object-oriented product data model specifications. They are structured in a deep inheritance graph and are intended to describe and exchange three-dimensional building and construction data in the Architecture, Engineering, Construction/Facilities Management industry (AEC/FM). The IFC schema is defined in EXPRESS, which is a data modeling language for product data based on ISO 10303-11. Various data types are defined within its schema, which are explained below:

- Simple data types are String, Binary, Logical, Boolean, Number, Integer and Real. They are used to specify the data type of an attribute within more complex types.
- Defined data types can be identified by using the keyword *TYPE*. It is used to define other types, e.g., to define a data type, which represents a geometrical dimension greater than zero and less than or equal to three.

```
TYPE IfcDimensionCount = INTEGER;
  WHERE
    WR1 : { 0 < SELF <= 3 };
END_TYPE;
```

- Enumeration data types that are a special type of defined data type can be identified by using the keyword *ENUMERATION OF*. They contain simple strings as values.

```
TYPE IfcProfileTypeEnum = ENUMERATION OF
  (CURVE, AREA);
END_TYPE;
```

- Select data types that are also a special type of defined data type can be identified by using the keyword *SELECT*. They are used to define choices or alternatives between different options and can consist of defined types, as well as entity types.

```
TYPE IfcAxis2Placement = SELECT
  (IfcAxis2Placement2D, IfcAxis2Placement3D);
END_TYPE;
```

- Entity data types that are similar to classes in the object-oriented software paradigm can be identified by using the keyword *ENTITY*. They can be related to an inheritance tree as either a subtype/supertype, or by attributes. Subtype entities can be identified

by using the keyword *SUBTYPE*, whereas supertype entities can be identified by using the keyword *SUPERTYPE*. In addition, entity data types can be marked as an abstract data type by the keyword *ABSTRACT* in order to ensure that an instance cannot be constructed. Furthermore, if they are related by attributes, then they are required, except when they are marked by the keyword *"OPTIONAL"*.

```
ENTITY IfcStructuralAction
  ABSTRACT SUPERTYPE OF (ONEOF
    (IfcStructuralLinearAction
    ,IfcStructuralPlanarAction
    ,IfcStructuralPointAction))
  SUBTYPE OF (IfcStructuralActivity);
    DestabilizingLoad : BOOLEAN;
    CausedBy : OPTIONAL IfcStructuralReaction;
END_ENTITY;
```

- Aggregation data types can be identified by using the keywords *LIST*, *ARRAY*, *SET*, *BAG*. They may consist of any other data types. In the case of a list/array, the members are stored in a specific order, whereas set/bag are stored in a non-specific order.

```
ENTITY IfcVirtualGridIntersection;
  IntersectingAxes : LIST [2:2] OF UNIQUE IfcGridAxis;
  OffsetDistances : LIST [2:3] OF IfcLengthMeasure;
END_ENTITY;
```

In addition to the defined data types, EXPRESS supports the specialization of defined data types, as well as entity data types. This is achieved by constraints, which can be defined through the keyword *WHERE*. Constraints must be fulfilled, otherwise the generated instance is not valid according to the EXPRESS schema. For example, to generate an *IfcLine* in a consistent manner, the dimension of the location point and the direction vector must be equal.

```
ENTITY IfcLine
  SUBTYPE OF (IfcCurve);
    Pnt : IfcCartesianPoint;
    Dir : IfcVector;
  WHERE
    WR1 : Dir.Dim = Pnt.Dim;
END_ENTITY;
```

8.2.2 APDL Script Schema

The simulation software ANSYS is widely applied in domains, such as structural mechanics, fluid mechanics, thermodynamics, acoustics, piezoelectricity, and electromagnetism. It is used in order to perform various types of analyses, such as modal analysis, stress analysis, thermal and fluid analysis, and coupled field analysis. The modeling and evaluation of engineering problems, including the handling of pre/post-processes is achieved by its Ansys Parametric Design Language (APDL).

The parametric language APDL enables the user to handle, automate and monitor the pre/post-processing via a set of commands. The commands can be executed directly via an internal command prompt, or indirectly via a separate text file known as a macro. Hence, the APDL can be understood as both a scripting language and a macro language. The macro text file is a sequence of commands that are collected in order to perform specific tasks in a general way, e.g., for geometrical modeling. Macros enable the creation of custom ANSYS commands. Furthermore, macros can be nested, which means that one macro can call a second macro, and the second macro can then call a third one, etc.

An ADPL command can be simply executed through the command's name and a sequence of expressions. Expressions can be numerical values and alphanumeric character strings, e.g., to identify names of parameters, entity keywords, labels, or to add specific values.

```
CMD_Name, Exp1, Exp2, Exp3, Exp4, Exp5, ...
```

The number and type of expressions depend on the command used. For example, three commands are outlined, which are needed for modeling extruded areas.

- The *K* command defines a keypoint by a reference number *NPT* and a location *X*, *Y*, *Z*.

```
K, NPT, X, Y, Z
```

- The *A* command defines an area by a list of connecting keypoints *P1, P2, ..., P18*.

```
A, P1, P2, P3, ... , P18
```

- The *VEXT* command generates additional volumes by extruding areas through a set of areas *NA1,NA2,NINC*, increments *DX,DY,DZ* and scale factors *RX,RY,RZ*.

```
VEXT, NA1, NA2, NINC, DX, DY, DZ, RX, RY, RZ
```

8.3 Schema Analysis

Data structures inside of a schema $S \in Q$ can be interrelated to describe more complex facts. These relationships are included in the power set $P(S)$. From a mathematical point of view, the power set contains all the possible relations corresponding to the single data structures of a schema. However, from a software-engineering point of view, the data structures of a schema S are only in a relationship with a subset of data structures $T \subseteq P(S)$.

The schema analysis is used to figure out all the logical connections in $P(S)$ which, from a software-engineering point of view, are not included in the schema S. Those relations will never be activated during the mapping and evaluation process, therefore they can be removed from the original power set. Hence, a schema analysis has to take the various data types into account (e.g., abstract data types) and the different types of logical connections (e.g., general relations, instance/class-level relations and optional/required relations).

The IFC has been applied in domains, such as construction scheduling, change management, design and construction, HVAC engineering, cost estimating, and facilities management. This is achieved through more than 600 strongly interrelated entity data types ($IFC2x3$), which are ingrained in a deep inheritance tree. However, this leads to an enormous number of combinations in $P(IFC2x3)$, which have to be considered during the mapping and evaluation process. Hence, schema analyses have been carried out only on the subsets of entities.[1] They include only the essential entities for modeling the primary and secondary building elements, the material properties, and the structural elements of the frame structure.

The schema analyses are performed stepwise by a self-developed JAVA tool. It takes the different data types and logical connections introduced in chapter 7 into account. The steps can be classified according to the following three main categories:

- *Schema rule* steps are based on the schema itself, i.e., data types, logical connections, and constraints. They can be applied to every other subset derived from the schema.
- *Global modeling* steps are based on the assumptions made in a global context, e.g., positioning of elements. They may be applied to other subsets as well.
- *Local modeling* steps are based on the assumptions made on the direct relation to the subset and the task to be solved. They are rarely applicable to other subsets.

[1] This practice is propagated by buildingSMART as Model View Definitions (MVD).

Analysis of Primary Building Elements

The primary building elements of the frame structure are the column and beam. Within the IFC, they are embedded on the same level of the inheritance tree and defined as a subtype entity of *IfcBuildingElement* as shown in Figure 8.3. From an architectural point of view, the differences as defined in the documentation [35] are the following:

- *"An IfcBeam is a horizontal, or nearly horizontal, structural member."*
- *"An IfcColumn is a vertical structural member which often is aligned with a structural grid intersection. It represents a vertical, or nearly vertical, structural member."*

From a software engineering point of view, *IfcColumn* and *IfcBeam* are identical in regards to their supertype entities and inherited attributes. Hence, a schema analysis for *IfcColumn* and *IfcBeam* would lead to identical combinations in the power set. In the following, a schema analysis for a chosen subset $S_{Col} \subseteq IFC$ has been performed. The subset contains all the essential entities for modeling columns. Due to their equality, the result can be used for modeling beams as well.

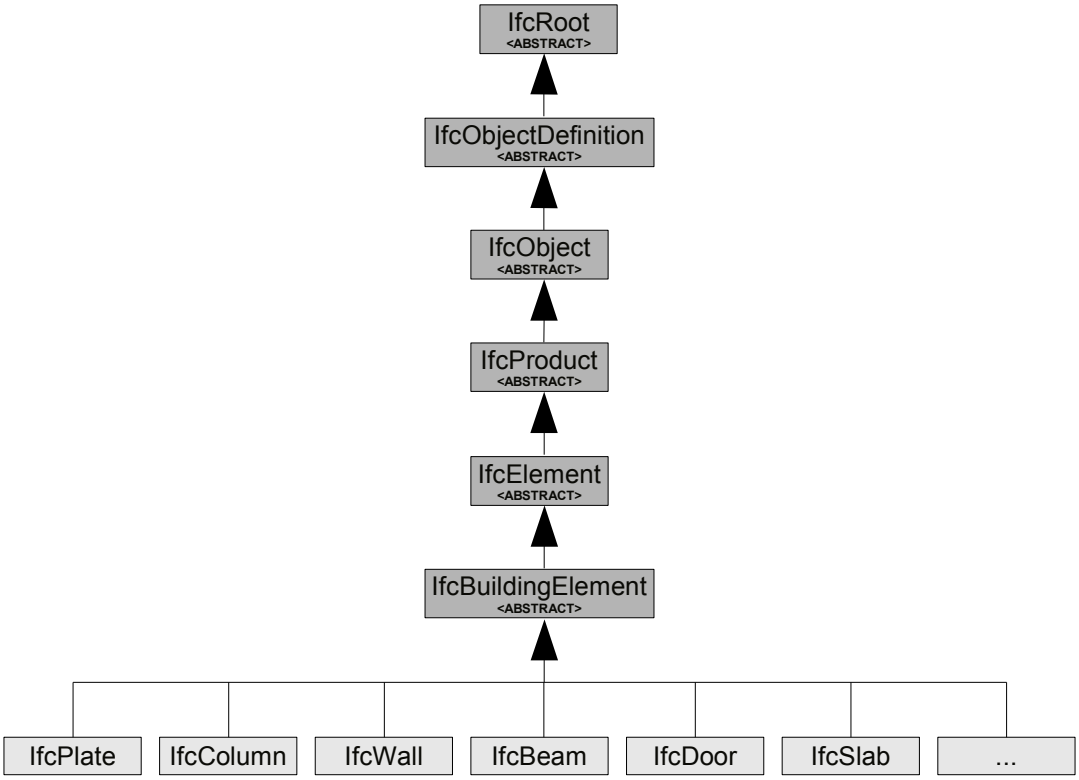


Figure 8.3: Inheritance tree of *IfcBuildingElement*.

The subset S_{Col} comprises 56 more or less significant entities for modeling columns which is a primary element of the frame structure. The modeling is achieved by entity *IfcColumn* and all of its interrelated entities. A simplified, incomplete aggregation and inheritance graph with regards to *IfcColumn* is shown in Figure 8.4.

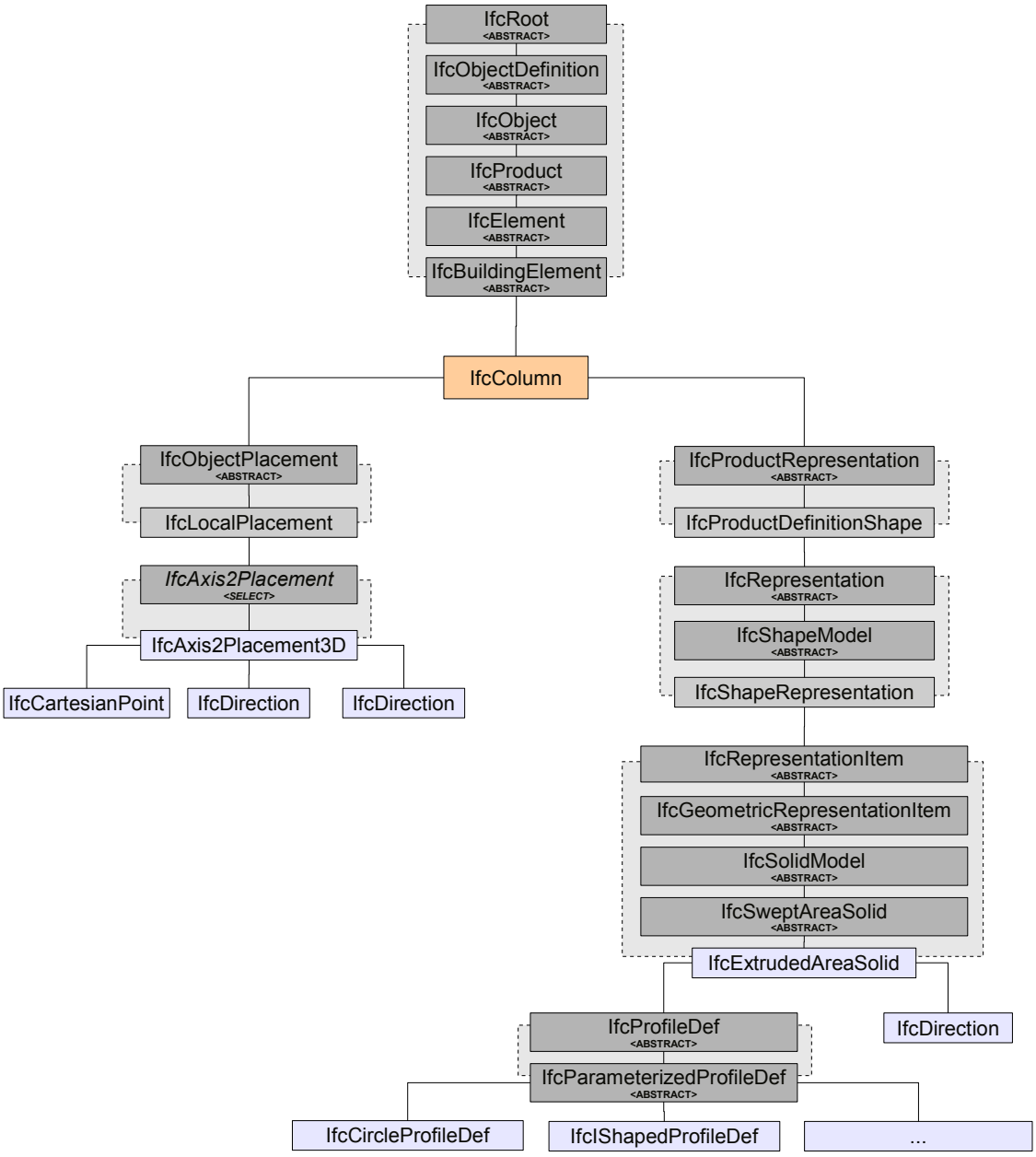


Figure 8.4: The essential IFC entities for modeling columns.

The power set of S_{Col} mathematically leads to $|P(S_{Col})| = 2^{56}$ combinations, which are more or less valid according to the IFC schema definition. To figure out all the invalid combinations and reduce the number of combinations, a stepwise schema analysis has been carried out:

- The schema rule steps, including abstract data types, relationships, and constraints:
  - Step 1: All combinations containing invalid relationships have been removed. The number of combinations has been reduced to $|P(S_{Col})| = 159165$.
  - Step 2: All combinations missing the required entity attributes have been removed. The number of combinations has been reduced to $|P(S_{Col})| = 7085$.
  - Step 3: All the columns that include a product representation, but no object placement are invalid (WHERE rule) and have been removed. The number of combinations has been reduced to $|P(S_{Col})| = 6528$.
  - Step 4: In future releases of the IFC, the two entities *IfcProductRepresentation* and *IfcProductRepresentation* will be changed into an ABSTRACT supertype. Hence, all combinations including one of the two entities have been removed. The number of combinations has been reduced to $|P(S_{Col})| = 1632$.
  - Step 5: Due to the fact that the modeling of columns is in the three-dimensional space, all combinations that use two-dimensional object placements have been removed. The number of combinations has been reduced to $|P(S_{Col})| = 1088$.
  - Step 6: All the combinations, which include only one of the two *IfCDirection* with regard to *IfcAxis2Placement3D* are invalid (WHERE rule) and have been removed. The number of combinations has been reduced to $|P(S_{Col})| = 288$.
- Global modeling steps are based on restrictions regarding the placement of columns:
  - Step 7: By using absolute placements instead of relative placements, the number of combinations has been reduced to $|P(S_{Col})| = 144$.
  - Step 8: To locate and originate an object and/or to define a placement coordinate system, a point and direction vectors are required.
    a: The placement of a column is achieved by *IfcLocalPlacement* and thus, both direction vectors in *IfcAxis2Placement3D* are required. Hence, all the combinations where *IfcAxis2Placement3D* does not contain any direction vectors have been removed. The number of combinations has been reduced to $|P(S_{Col})| = 72$.

b: An additional placement of the solid representation in *IfcExtrudedAreaSolid* is not necessary. Hence, all the combinations where *IfcAxis2Placement3D* contains direction vectors have been removed. The number of combinations has been reduced to $|P(S_{Col})| = 40$.

c: An additional placement of the area representation in *IfcProfileDef* is not required. Hence, all the combinations where *IfcAxis2Placement2D* contains a direction vector have been removed. The number of combinations has been reduced to $|P(S_{Col})| = 24$.

- Local modeling steps are based on restrictions with respect to geometrical modeling:
  - Step 9: Within the frame structure, the columns are modeled as extruded area solids only. Hence, all the columns that include another representation item have been removed. The number of combinations can be reduced to $|P(S_{Col})| = 16$.
  - Step 10: Within the frame structure, only rectangular, circular and I-shaped columns are needed. Hence, all the combinations which use other shape profiles have been removed. The number of combinations can be reduced to $|P(S_{Col})| = 3$.

The influence of the stepwise schema analysis in regards to the number of valid combinations in $P(S_{Col})$ is summarized in Figure 8.5.

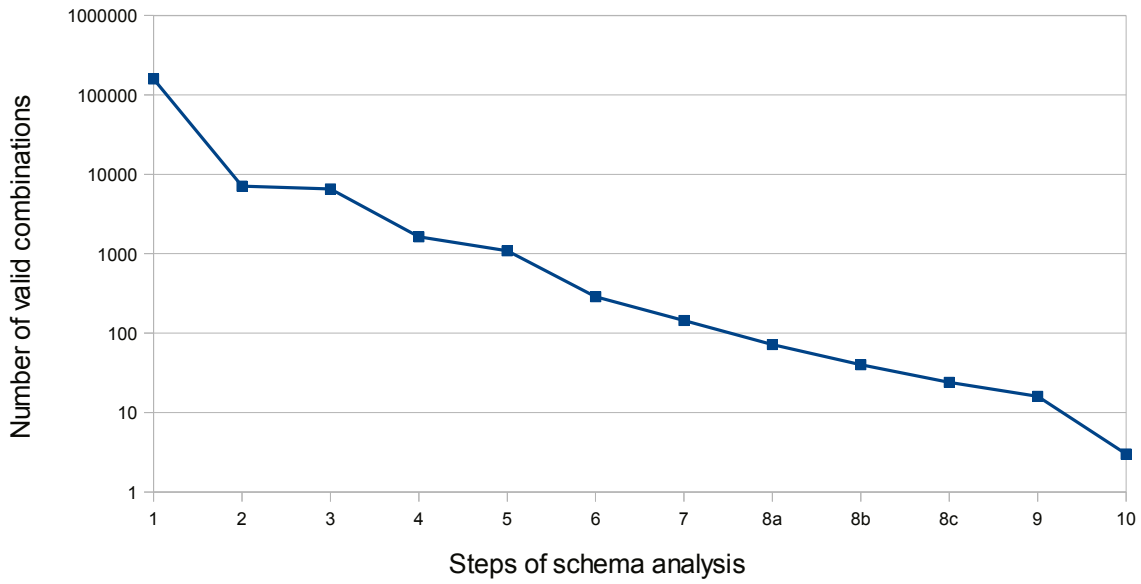


Figure 8.5: The influence of the stepwise schema analysis.

Final Results

The power set $P(S_{Col})$, has finally been reduced to only three combinations. Each of them includes the minimal set of IFC entities for modeling a certain type of column (Figure 8.6).

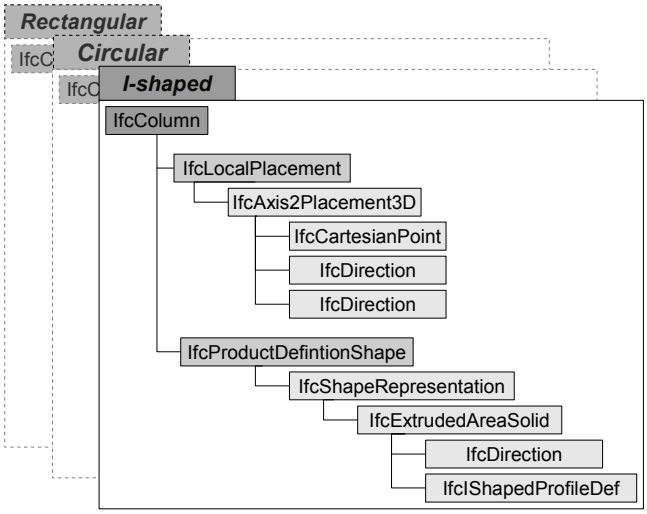


Figure 8.6: The minimal relationship tree for modeling I-shaped columns.

As established on page 106, *IfcColumn* and *IfcBeam* are identical with respect to their entity definitions. A schema analysis for *IfcBeam* will therefore lead to identical results (Figure 8.7).

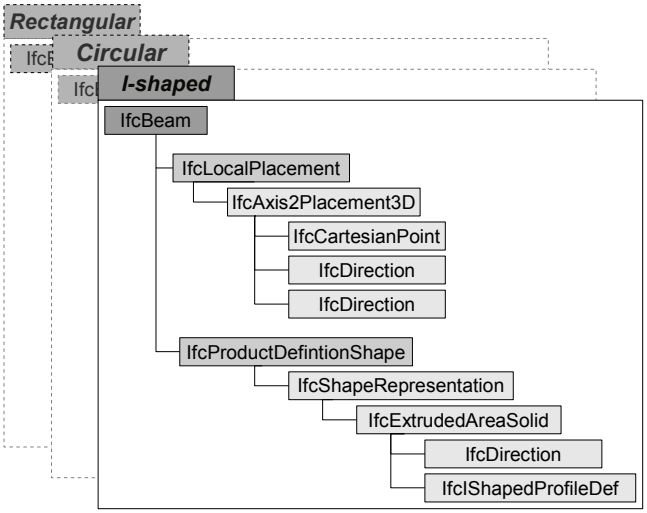


Figure 8.7: The minimal relationship tree for modeling I-shaped beams.

Finally, the number of combinations for modeling the primary building elements is six, three of which are of entity type *IfcColumn* and the others of entity type *IfcBeam*.

8.4 Schema Mapping

Schema mapping is one of the key issues within the proposed approach and provides the basis for assessing the data exchange. The mapping of schema coupling $(IFC, S_{ANSYS}) \in C$ is achieved with the schema mapping $m_{IFC,S_{ANSYS}}$. It maps the IFC entity data types to corresponding ANSYS APDL macros/commands:

$$m_{IFC,S_{ANSYS}} : P(IFC) \rightarrow P(S_{ANSYS}) \text{ with } IFC, S_{ANSYS} \in Q$$

However, within the case study, only subsets of the IFC are considered, e.g., for the modeling of primary and secondary building elements, materials, or structural elements. The related schema mapping for subset $S_{Col} \subseteq IFC$ is:

$$m_{S_{Col},S_{ANSYS}} : P(S_{Col} \subseteq IFC) \rightarrow P(S_{ANSYS})$$

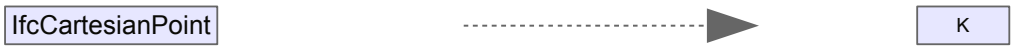
In addition, the power set $P(S_{Col})$ could be reduced via schema analysis (section 8.3) to only three different elements, which have to be mapped. The mapping of the primary building elements can be divided into a mapping of their individual local placement, as well as a mapping of their geometrical representation.

Local Placement

The placement of primary building elements is achieved through a local coordinate system. Within the IFC, it is defined by a location point and two direction vectors, whereas in ANSYS, it is defined by three points (Figure 8.8), as shown below:

Figure 8.8: Local coordinate systems: IFC (left) and ANSYS (right).

- The location point $P_{location}$ can be mapped directly from the IFC schema to the ANSYS schema. Within the IFC, it is modeled through the entity $IfcCartesianPoint$, whereas in ANSYS it is modeled through the K command.



- The information for defining the x-axis cannot be mapped directly. Within ANSYS, the point P_x is needed, which is modeled through the K command. Its coordinates can be computed by adding up the location point $P_{location}$ ($IfcCartesianPoint$) and the corresponding direction $\vec{x}$ ($IfcDirection$).

- The information for defining the y-axis cannot be mapped directly. Within ANSYS, the point P_y is needed, which is modeled through the K command. Its coordinates can be computed through the cross product of both directions $\vec{x} \times \vec{z}$.

The complete mapping of the local placement is shown in Figure 8.9. Within the IFC schema, the local placement (*IfcLocalPlacement*) is defined by the entity *IfcAxis2Placement3D*. It has to be mapped to the corresponding *CSKP* command. During the mapping, various conversions are needed, which may reduce the quality of the mapping.

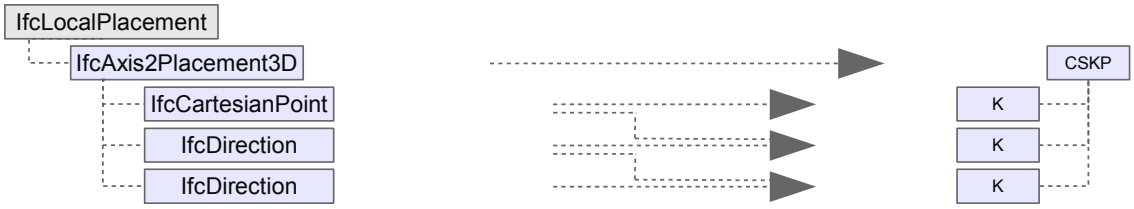


Figure 8.9: The mapping of the local placement.

Geometrical Representation

The geometrical representation of primary building elements is achieved through extruded area solids. Within the IFC, the areas are parameterized, whereas in ANSYS they are defined as a polygon (Figure 8.10).

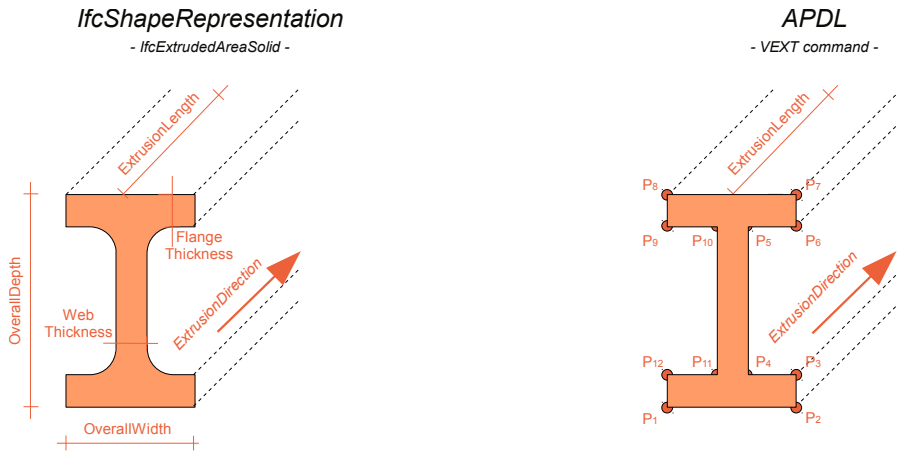


Figure 8.10: Geometrical representation: IFC (left) and ANSYS (right).

- Within the IFC schema, the extrusion of an area can be modeled through the entity *IfcExtrudedAreaSolid* plus its extrusion direction (*IfcDirection*). Within ANSYS, it is modeled by the *VEXT* command.

- The area which has to be extruded can be described using parametrized profile entities, e.g., *IfcIShapeProfileDef*, *IfcCircleProfileDef*, or *IfcRectangleProfileDef*. In contrast, the area within ANSYS is modeled as a polygon through the *A* command.

The complete mapping of the geometrical representation is shown in Figure 8.11. Within the IFC schema, the geometry of a product is defined by the entity *IfcProductDefinitionShape*. It includes a specific parametrized shape definition (*IfcShapeRepresentation*), which has to be extruded. Within ANSYS, this fact is modeled through an extrusion of polygonal areas via the *VEXT* command.

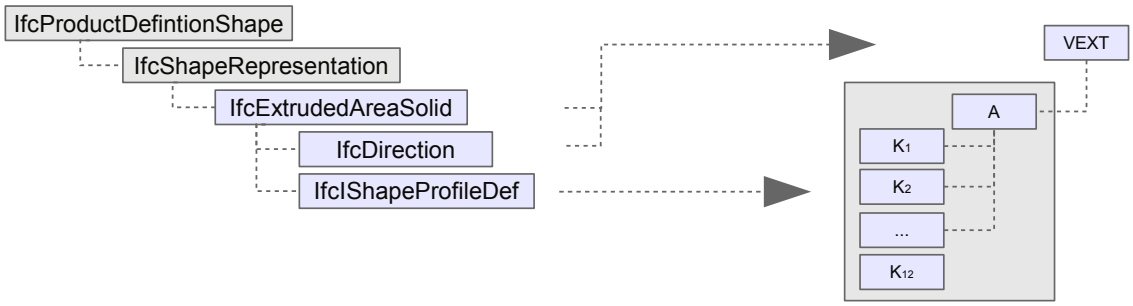


Figure 8.11: The mapping of the geometrical representation.

Final Results

The primary building elements of the frame structure has been modeled using ten interrelated IFC entities. In order to exchange these, they can be mapped to an equivalent ANSYS APDL macro consisting of 18 more or less nested command calls. Figure 8.12 shows the mapping of an I-shaped column $\in P(S_{Col})$.

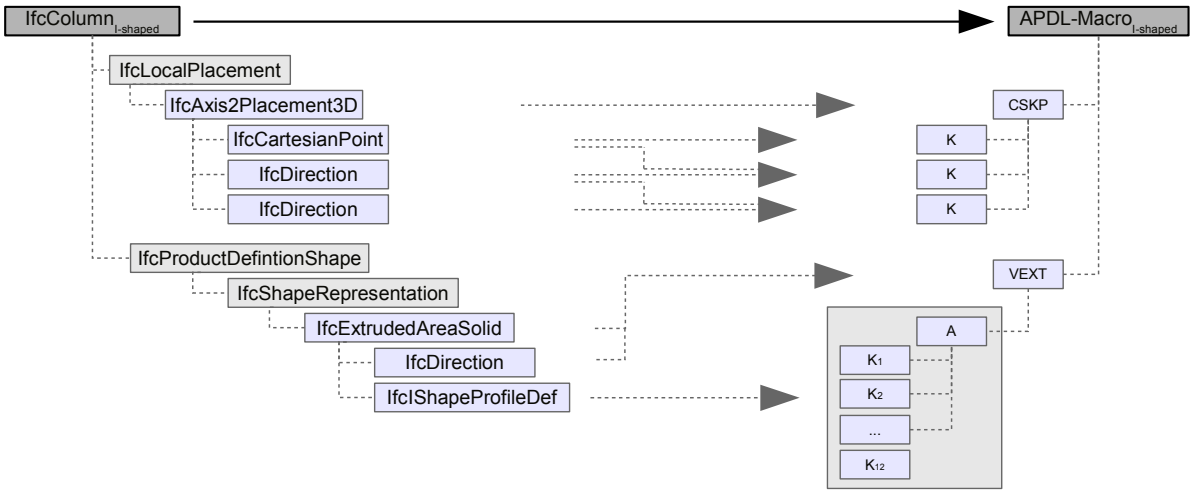


Figure 8.12: The schema mapping of an I-shaped column.

Figure 8.13 shows the results of schema mapping for certain I-shaped building element instances. However, various conversions within the mapping process are made, which may reduce the quality of data exchange. They have been taken into account within the evaluation process of section 8.5.

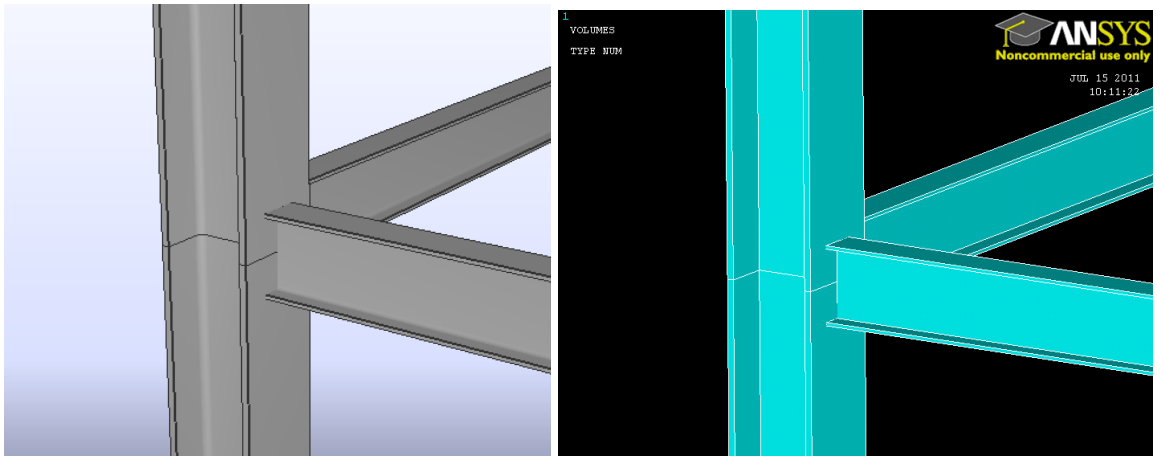


Figure 8.13: Differences in modeling I-shaped columns: IFC (left) and ANSYS (right).

8.5 Evaluated Schema Mapping

The range of quality values R has been chosen as the interval $[x_{min}, x_{max}]$. The lower bound x_{min} describes the worst case scenario and the upper bound x_{max} describes the best case scenario of a certain mapping. A value of *1* represents no loss of information, whereas a value of *less than 1* means that only a part of the information can be exchanged:

$$R := \{(x_{min}, x_{max}) \in \mathbb{R} \times \mathbb{R} \mid 0 < x_{min} \leq 1 \wedge 0 < x_{max} \leq 1 \wedge x_{min} \leq x_{max}\}$$

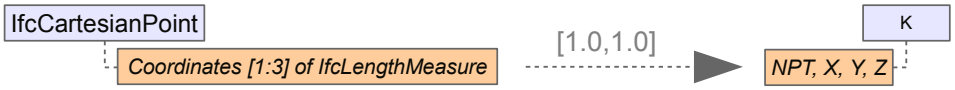
Finally, each mapping of the data structures has to be associated with such a quality interval. An assessment of schema mapping $m_{S_{Col},S_{ANSYS}}$ which maps the different types of columns is achieved by an evaluated schema mapping:

$$\bar{m}_{S_{Col},S_{ANSYS}} : m_{S_{Col},S_{ANSYS}} \rightarrow R$$

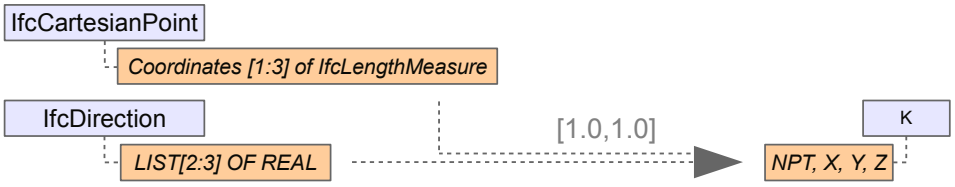
The ascertainment of quality values has to be determined based on the attributes of the data structures that have to be mapped.

Local Placement

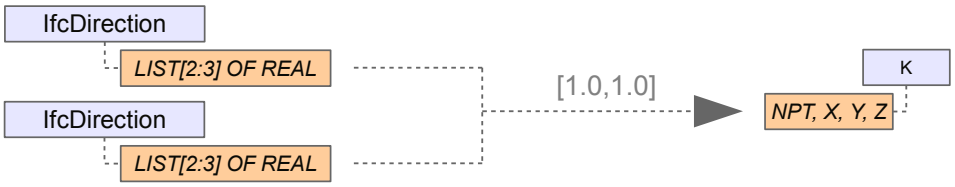
- The location point is modeled through the K command. Its coordinates x, y, and z can be mapped directly from $Coordinates$, which is an attribute of $IfcCartesianPoint$, which contains the coordinates. The mapping is lossless $[1.0, 1.0]$.



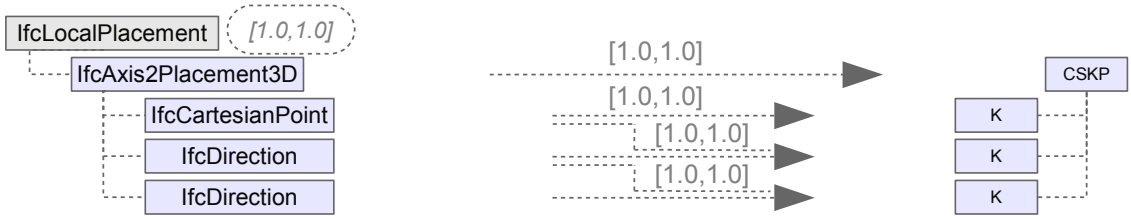
- The x-axis is defined by P_x, which is modeled through the K command. Its coordinates cannot be mapped directly. They have to be computed by adding up the location point coordinates included in $IfcCartesianPoint$ and its direction values included in $IfcDirection$. The mapping is lossless $[1.0, 1.0]$.



- The y-axis is defined by P_y, which is modeled through the K command. Its coordinates cannot be mapped directly. They have to be computed through the cross product of the x-axis included in $IfcDirection$ and the z-axis included in $IfcDirection$: $\vec{x} \times \vec{z}$. The mapping is lossless $[1.0, 1.0]$.



Finally, the mapping of *IfcLocalPlacement* is lossless since all of its child branches are lossless:



Geometrical Representation

- The extrusion of an area is modeled through the *VEXT* command. On the condition that an extrusion occurs only in relation to one axis, the extrusion depth DX, DY, DZ can be mapped directly from $Depth$, which is an attribute of $IfcExtrudedAreaSolid$. The mapping is lossless $[1.0, 1.0]$.

- The extrusion area is modeled through the *A* command. It defines a polygonal area through a set of key points. However, its points cannot be mapped directly because of a parametrized area definition within the IFC schema. In the case of modeling I-shaped building elements, this inevitably leads to geometrical approximation errors, which can be estimated on the basis of differences in its surface areas (Figures 8.14 and 8.15).

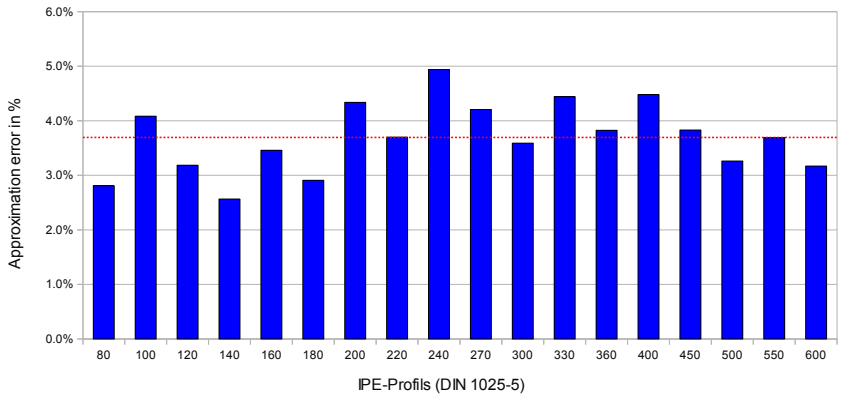


Figure 8.14: The geometrical approximation error of IPE profiles.

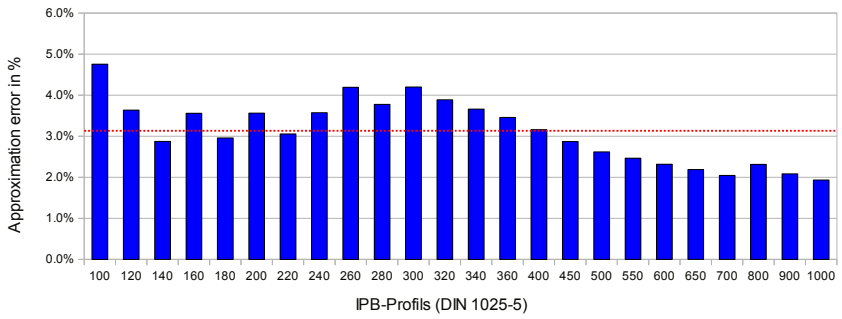
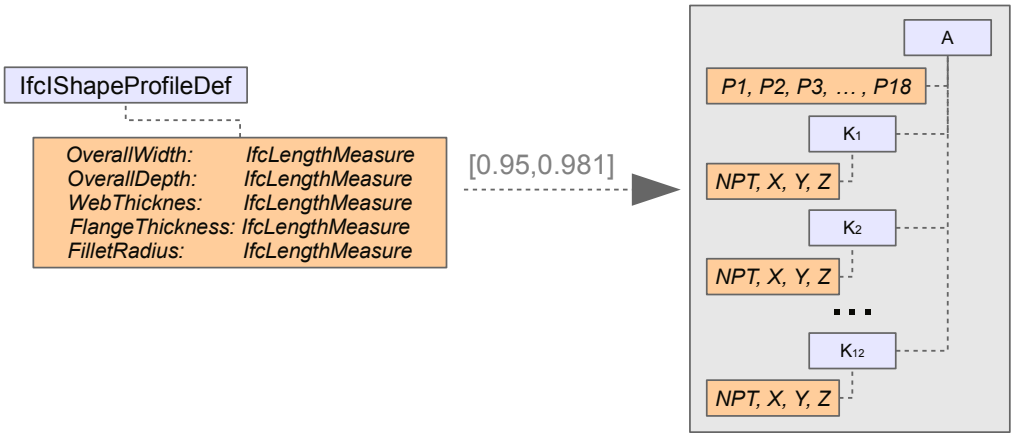


Figure 8.15: The geometrical approximation error of IPB profiles.

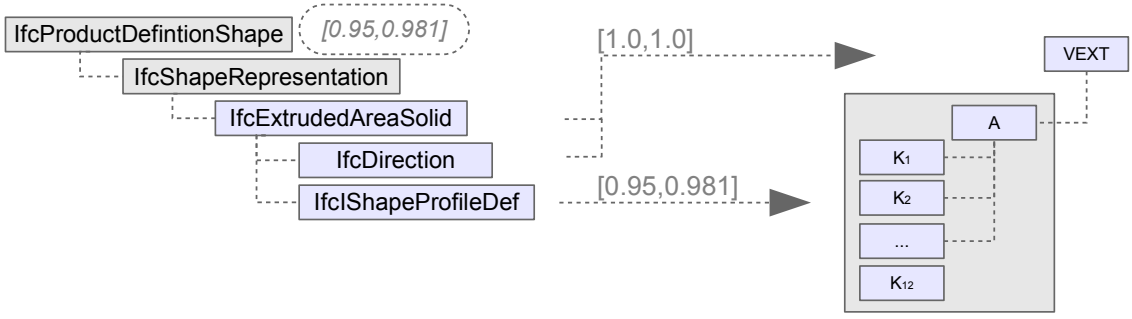
– The minimal geometrical error of 1.9%, which occurs for a mapping of IPB 1000 profiles (Figure 8.15), results in a mapping quality of $Q = 1.0 - 0.019 = 0.981$. It represents the best case x_{min} for mapping I-shaped building elements.

– The maximal geometrical error of 5.0%, which occurs for a mapping of IPE 240 profiles (Figure 8.14), results in a mapping quality of $Q = 1.0 - 0.05 = 0.95$. It represents the worst case x_{max} for mapping I-shaped building elements.

The mapping of extrusion areas is lossy with regard to I-shaped profiles. The quality that can be expected, estimated according to the differences in its surface areas, has to be in the interval of $[0.95, 0.981]$.



Finally, the mapping of the geometrical representation is lossy with regard to I-shaped area profiles since one of its child branches (*IfcIShapeProfileDef*) is lossy. The quality for mapping $IfcProductDefinitionShape$ can be computed through its partial qualities for extrusion $[1.0, 1.0]$ and for modeling the area $[0.95, 0.981]$, which gives a result of $[0.95, 0.981]$:



Final Results

The quality for mapping primary building elements can be computed through its partial qualities $IfcLocalPlacement$ and $IfcProductDefinitionShape$. It is exemplified in Figure 8.16 for I-shaped columns. The mapping quality of its local placement is lossless $[1.0, 1.0]$, whereas the mapping quality of its geometrical representation is lossy $[0.95, 0.981]$. The final quality gives a result of $[0.95, 0.981]$.

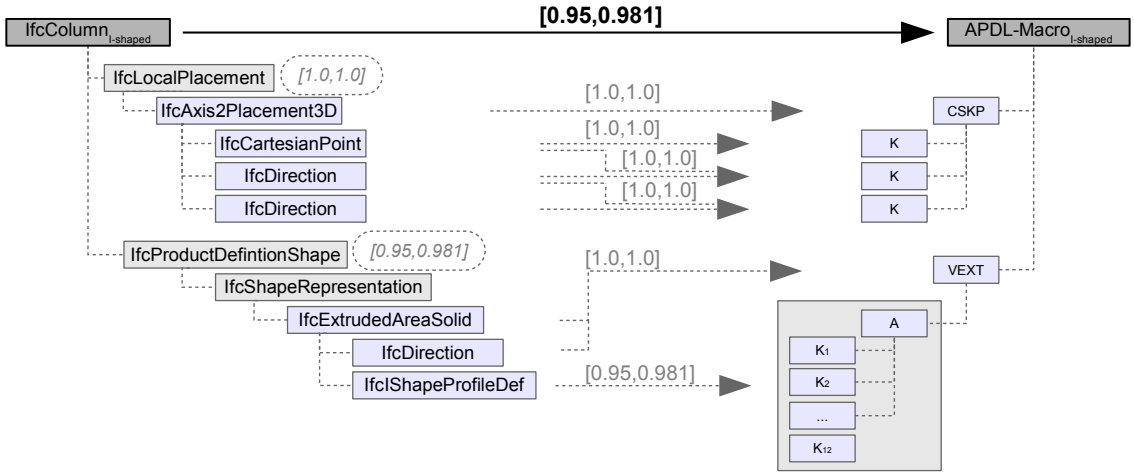


Figure 8.16: The mapping quality of I-shaped columns.

As established on page 106, *IfcColumn* and *IfcBeam* are identical. Hence, the evaluation process for $IfcBeam$ would lead to identical results. Finally, the evaluated mapping of primary building elements is shown in Table 8.1.

$\bar{m}_{S_{Pri},S_{ANSYS}}$	M_{COL_R}	M_{COL_I}	M_{COL_C}	M_{BEAM_R}	M_{BEAM_I}	M_{BEAM_C}	$\varnothing$
IFC_{COL_I}	[0,0]	[0.95,0.981]	[0,0]	[0,0]	[0,0]	[0,0]	-
IFC_{COL_R}	[1,1]	[0,0]	[0,0]	[0,0]	[0,0]	[0,0]	-
IFC_{COL_C}	[0,0]	[0,0]	[1,1]	[0,0]	[0,0]	[0,0]	-
IFC_{BEAM_I}	[0,0]	[0,0]	[0,0]	[0,0]	[0.95,0.981]	[0,0]	-
IFC_{BEAM_R}	[0,0]	[0,0]	[0,0]	[1,1]	[0,0]	[0,0]	-
IFC_{BEAM_C}	[0,0]	[0,0]	[0,0]	[0,0]	[0,0]	[1,1]	-
$\varnothing$	-	-	-	-	-	-	[1,1]

Table 8.1: The evaluated mapping $\bar{m}_{S_{Pri},S_{ANSYS}}$.

8.6 Quality Assessment

The quality of data exchange for arbitrary sets of instances I_A from schema A to another schema B can be computed in an a priori manner on the basis of an evaluated schema mapping $\bar{m}_{AB}$ (page 75):

$$Quality := f(\bar{m}_{AB}, I_A) := \sum_{i \in P(I_A)} \sum_{j \in P(B)} \frac{\bar{m}_{AB}(type(i), j)}{n_{type(i)}}$$

8.6.1 Scenario 1

The quality of data exchange for primary building elements has to be computed. The instance set I_1 as shown in Figure 8.17 consists of 30 column and 35 beam instances. They are defined by the IFC schema and have to be transferred to ANSYS. The corresponding evaluated schema mapping is $\bar{m}_{S_{Pri},S_{ANSYS}}$ (Table 8.1). The overall quality of data exchange can be computed a priori as follows:

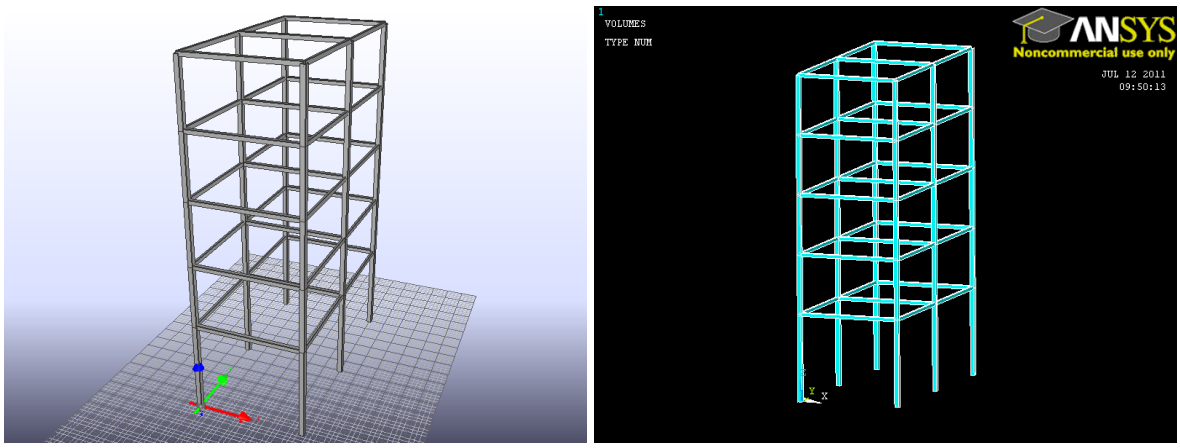


Figure 8.17: The frame structure modeled within the IFC (left) and ANSYS (right).

- Without having any details regarding the different types of primary elements used, the quality has to be in the range from 0.95 for I-shaped profiles (worst case) to 1.0 for rectangular and circular profiles (best case). Hence, the expected quality must be in the following interval:

$$Quality = \frac{30 \cdot [0.95, 1] + 35 \cdot [0.95, 1]}{30 + 35} = [0.95, 1]$$

- A more precise quality can be estimated by taking the different types of building elements used into account. The frame structure has been modeled only through I-shaped profiles. The quality has to be in the range from 0.95 for IPE 240 (worst case) to 0.981 for IPB 1000 (best case). Hence, the quality must be in the following interval:

$$Quality = \frac{30 \cdot [0.95, 0.981] + 35 \cdot [0.95, 0.981]}{30 + 35} = [0.95, 0.981]$$

Finally, the frame structure consists of 65 I-shaped element instances, of which 30 are IPE 200 with an exchange quality of 0.9566, 5 are IPB 200 with an exchange quality of 0.9644, and 30 are IPB 240 with an exchange quality of 0.9643. The final quality of data exchange is as follows:

$$Quality = \frac{30 \cdot 0.9566 + 5 \cdot 0.9644 + 30 \cdot 0.9643}{30 + 5 + 30} = 0.9608$$

8.6.2 Scenario 2

The quality of data exchange for primary and secondary building elements has to be computed. The instance set I_2 as shown in Figure 8.18 consists of 65 primary elements (30 columns, 35 beams) and 58 secondary elements (48 bracings, and 10 slabs). They are defined by the IFC schema and have to be transferred to ANSYS.

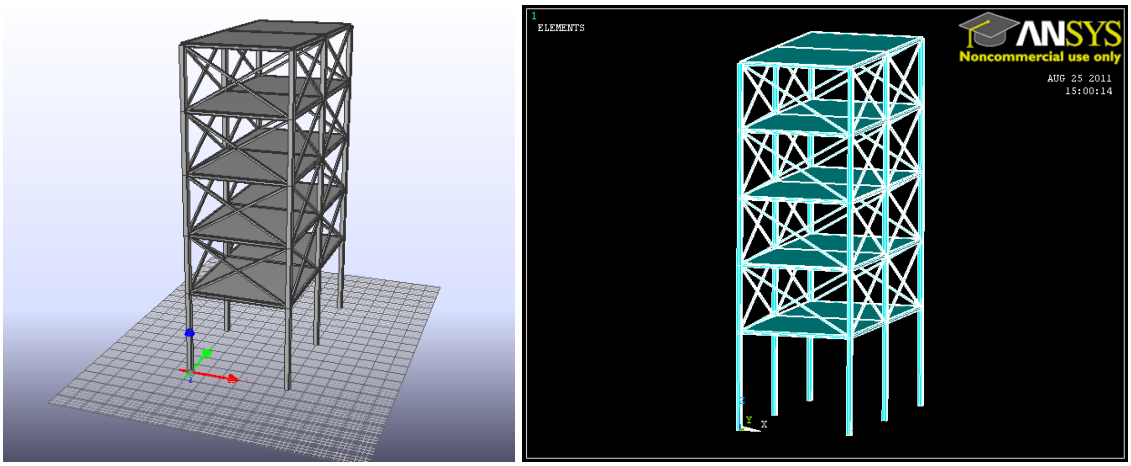


Figure 8.18: The frame structure modeled within the IFC (left) and ANSYS (right).

The evaluation of secondary building elements has been performed in exactly the same way as for primary elements. A bracing, which is defined by the IFC through the entity *IFCMember*, has been mapped to the ANSYS APDL macro M_{LINK}. A slab, which is defined by the

IFC through the entity *IFCSlab* has been mapped to the ANSYS APDL macro M_{SHELL}. The results of the mappings are shown in Table 8.2 below:

$\bar{m}_{S_{Sec},S_{ANSYS}}$	M_{SHELL}	M_{LINK}	$\varnothing$
IFC_{SLAB}	[1,1]	[0,0]	-
IFC_{MEMBER}	[0,0]	[1,1]	-
$\varnothing$	-	-	[1,1]

Table 8.2: The evaluated mapping $\bar{m}_{S_{Sec},S_{ANSYS}}$.

The evaluated schema mappings for computing an exchange quality of the primary and secondary building elements are $\bar{m}_{S_{Pri},S_{ANSYS}}$ (Table 8.1) and $\bar{m}_{S_{Sec},S_{ANSYS}}$ (Table 8.2). The overall quality of data exchange can be computed as follows.

- Without having any details regarding the different types of primary and secondary elements used, the quality has to be in the following interval:

$$Quality = \frac{30 \cdot [0.95, 1] + 35 \cdot [0.95, 1] + 48 \cdot [1, 1] + 10 \cdot [1, 1]}{30 + 35 + 48 + 10}$$

$$Quality = [0.9736, 1.0]$$

- A more precise quality can be estimated by taking the different types of building elements used into account. The frame structure has been modeled only by I-shaped columns and beams, rectangular slabs and geometry-free bracings. Hence, the quality must be in the following interval:

$$Quality = \frac{30 \cdot [0.95, 0.981] + 35 \cdot [0.95, 0.981] + 48 \cdot [1, 1] + 10 \cdot [1, 1]}{30 + 35 + 48 + 10}$$

$$Quality = [0.9736, 0.99]$$

Finally, the frame structure consists of 65 I-shaped element instances, of which 30 are IPE 200 with an exchange quality of 0.98915, 5 are IPB 200 with an exchange quality of 0.9911, and 30 are IPB 240 with an exchange quality of 0.9911; in addition, 48 geometry-free bracings have an exchange quality of 1.0; and 10 rectangular slabs with an exchange quality of 1.0. The final quality of data exchange is as follows:

$$Quality = \frac{30 \cdot 0.9566 + 5 \cdot 0.9644 + 30 \cdot 0.9643 + 48 \cdot 1 + 10 \cdot 1}{30 + 5 + 30 + 48 + 10} = 0.9793$$

9 Summary and Outlook

9.1 Summary

Software coupling is quite challenging due to the large number of incompatible software on the market. Semantic and technical decisions must be made in order to find adequate coupling strategies. Decisions are required to clarify *how software applications are coupled* and *what information has to be exchanged*. However, with respect to the numerous software technologies and paradigms available, a wide range of coupling aspects must be considered, and decisions are made accordingly. Another concern is related to the examination of *how well the software applications can be coupled*, which may be used as an indicator of confidence in the output of coupled software applications. The quality assessment of coupled civil engineering applications has been an area of ongoing discussion in the research community in the last decade. However, performing these quality assessments is quite challenging because of the great quantity of proprietary data models that exist, which are developed by vendors, organizations and consortia. Data interoperability plays an important role for the software coupling used in the construction industry. Therefore, the research objectives of this study are related to both software coupling and its quality assessment.

In this study, a coupling graph is introduced, which includes the various coupling aspects. It supports the engineers in the development or selection of adequate coupling strategies. The proposed coupling graph is flexible in its depth and breadth, which depends on the number of coupling aspects and software technologies considered. This makes it possible to create coupling graphs according to the different levels of experience and knowledge of the software engineers. Furthermore, integration of the coupling graph into a four-layered coupling architecture guided the development of the coupling pattern language. It enabled the description of coupling strategies at different levels of abstractions. They range from abstract coupling templates via coupling patterns to specific coupling instances. Therefore, the knowledge of coupling strategies can be described in a reusable way, independent of the specific software and hardware requirements. The coupling graph approach is successfully applied to the coupling scenarios used in the research training group.

A posteriori approaches are widely used in practice in order to evaluate coupled systems. However, they are restricted in their scope and applicability. Therefore, an a priori approach based on the evaluated schema mapping was developed. This approach is dependent on the participating schemas and their mappings. The process of schema mapping and its evaluation is described using mathematical expressions derived from the set theory and graph theory by taking the various mapping patterns into account. Moreover, the coupling quality has been evaluated within the formalization process considering the uncertainties arising from the mapping process. Uncertainties are described using mathematical expressions derived from interval arithmetic. Finally, the evaluation process resulted in global quality values, which can be utilized by the user to assess the exchange. The quality values can be arbitrarily defined either linguistically or numerically. Due to the fact that data structures are interrelated, the proposed approach is applied to the basic principles of the object-oriented paradigm by taking certain types of relationships into account. The importance of schema analysis is shown for a case study in engineering. The applicability of the proposed evaluation schema mapping approach is shown for a coupling scenario of the research training group.

9.2 Outlook

Software development is a continuous process, which is influenced by the rapid evolution in computer science. Software technologies and paradigms are developed in short cycles. They are adopted by software to improve and extend its functionality. As a result, a large number of software concepts exist for ensuring the interoperability of frameworks, which is a requirement for coupling heterogeneous software systems. In addition, software applications may run on different operating systems and computers. They are also implemented by different programming languages. These languages are also based on different programming paradigms. With respect to the coupling of civil engineering applications, the following areas need to be researched further.

- Functional coupling, which means a sharing of functionality (e.g., processes, algorithms and methods) is briefly mentioned in this thesis. A further examination of its proper coupling aspects and software concepts, as well as its adaptation to the coupling graph and the meta-model architecture is needed.

- The object-oriented paradigm represents another challenge. Objects are used in software development to describe complex data structures. They consist of data fields (at-

tributes) and behaviors (methods, processes), which manipulate or process the data. Hence, it comprises properties of both functional and data coupling. Therefore, an examination of its proper coupling aspects and its adaptation to the coupling graph and the metamodel architecture is needed.

The design material in engineering practice (civil, mechanical and industrial engineering) is very sensitive. Lost or incorrect data can lead to numerous problems. An error-free exchange of digital data is one of the most important factors in achieving data interoperability and collaboration, and being able to distribute work efficiently across disciplines and organizations. The main focus in the last two decades has been on the development of PDT standards in order to achieve this goal. However, the exchange of data without errors and a loss of information by means of standardized data models is still not possible. Hence, data assessment approaches have become increasingly important for detecting inconsistencies and increasing the reliability of transferred data. With respect to a quality assessment of coupled civil engineering applications, the following areas need to be researched further.

- One challenge is how to combine existing a priori and a posteriori data assessment approaches into a centralized multistage analysis and assessment platform. Such an environment must be able to cope with the dynamic and distributed nature of the planning process. The combination of online software integration technologies, such as web-based, agent-based, or object-distributed systems, as well as centralized databases, will be a vital component of such a system.

- Another challenge is to improve and develop existing or new software quality metrics in order to ensure the consistency of data and to build confidence in the existing software and tools. Such quality metrics must be able to evaluate the data exchange for the task to be solved. For example, geometry and material data are more important for design and construction tasks when compared to facility management tasks. This can be achieved by employing weighting factors for the mapping of data structures.

- An additional area of research is data management. It includes transaction management, access control, version management, change notification/propagation mechanisms, etc. This becomes increasingly important because of the multiple phases of a project life cycle and the many multidisciplinary teams involved.

Bibliography

[1] *SIMVEC Berechnung und Simulation im Fahrzeugbau 2008 - SIMVEC Numerical Analysis and Simulation in Vehicle Engineering 2008 - 14. Internationaler Kongress und Fachausstellung*, volume 2031 of *VDI-Berichte/VDI-Tagungsbände*, 2008.

[2] *ASME Summer Bioengineering Conference (SBC)*, 2009.

[3] Graduiertenkolleg 1462. `http://www.uni-weimar.de/cms/bauing/forschung/grk1462.html`, 26. April 2011.

[4] ISO 15926. `http://15926.org`, 10. April 2011.

[5] Kiviniemi A., Fischer M., and Bazjanac V. Integration of multiple product models: Ifc model servers as a potential solution. Dresden, Germany, 2005. CIB W78 22nd Conference on Information Technology in Construction.

[6] AgentLink. `http://www.agentlink.org`, 26. April 2011.

[7] Brown Alex, Rezgui Yacine, Cooper Grahame, Yip Jim, and Brandon Peter. Promoting computer integrated construction through the use of distribution technology. *Journal of Information Technology in Construction (ITcon)*, 1(3):51 – 67, 1996.

[8] Christopher Alexander, Sara Ishikawa, and Murray Silverstein. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, New York, 1977.

[9] R. Amor. *A Generalised Framework for the Design and Construction of Integrated Design Systems*. PhD thesis, University of Auckland, Department of Computer Science, 1997.

[10] R. Amor. Supporting standard data model mappings. In *Proceedings of European Conference on Product and Process Modelling in the Building and Related Industries (ECPPM)*, pages 35 – 40, 2004.

[11] Inc. ANSYS. `http://www.ansys.com/`, 08. April 2011.

[12] Karola Antti, Lahtela Hannu, Hänninen Reijo, Hitchcock Rob, Chen Qingyan, Dajka Stephen, and Hagström Kim.

[13] G. Aouad. Integration: from a modelling dream into an implementation reality. In *Proceedings of CIB W55 (International Workshop on Information Support for Building Economics)*, pages 7 – 29, Lake District, UK, 1997.

[14] G. Aouad, M. Betts, P. Brandon, F. Brown, T. Child, G. Cooper, S. Ford, J. Kirkham, R. Oxma, M. Sarshar, and B. Young. Icon integration of construction information. Technical report, University of Salford [Department of Surveying and Information Technology Institute], 1994.

[15] G. Aouad and M. Sun. Information modelling and integration in the construction industry: a novel approach. *Structural Survey*, 17(2):82 – 88, 1999.

[16] G. Aouad, M. Sun, and I. Faraj. *Computer integrated construction: recent developments and future directions*, pages 27–53. Saxe-Coburg Publications, 2001.

[17] Graphisoft ArchiCAD. `http://www.graphisoft.de`, 26. April 2011.

[18] Autodesk Revit Architecture. `http://www.autodesk.de`, 26. April 2011.

[19] Bentley Architecture. `http://www.bentley.com`, 26. April 2011.

[20] Malcolm P. Atkinson, Misha Dmitriev, Craig Hamilton, and Tony Printezis. Scalable and recoverable implementation of object evolution for the pjama platform. In *Revised Papers from the 9th International Workshop on Persistent Object Systems*, POS-9, pages 292–314, London, UK, 2001. Springer-Verlag.

[21] B., Fang Z., Sun W., Shokoufandeh A., and Regli W. Starly. Three-dimensional reconstruction for medical-cad modeling. *Computer-Aided Design & Applications*, 2:431–438, 2005.

[22] Susanne Backas. Spadex - final report. `http://vera.vtt.fi/Documents/documents.htm`, 2001.

[23] N. Bakis, G. Aouad, and M. Kagioglou. Towards distributed product data sharing environments – progress so far and future challenges. *Automation in Construction*, 16(5):586 – 595, 2007.

[24] Jay Banerjee, Won Kim, Hyoung-Joo Kim, and Henry F. Korth. Semantics and implementation of schema evolution in object-oriented databases. *SIGMOD Rec.*, 16:311–322, December 1987.

[25] V. Bazjanac. Early lessons from deployment of ifc compatible software. In *European Community Product and Process Modeling Conference (ECPPM 2003)*, Portoroz, Slovenia, September 2002.

[26] Björk B.C. The ratas project - an example of co-operation between industry and research towards computer integrated construction. *ASCE Journal of Computing in Civil Engineering*, 8(4):401 – 419, 1994.

[27] Björk B.C. Ratas, a longitudinal case study of an early construction it roadmap project. *Journal of Information Technology in Construction (ITcon)*, 14(Special Issue Next Generation Construction IT: Technology Foresight, Future Studies, Roadmapping, and Scenario Planning):385 – 399, 2009.

[28] Kent Beck and Ward Cunningham. Using pattern languages for object oriented programs. In *Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, 1987.

[29] Djamal Benslimane, Schahram Dustdar, and Amit Sheth. Services mashups: The new generation of web applications. *IEEE Internet Computing*, 12:13 – 15, September 2008.

[30] A. Bijnen. Operation mapping or how to get the right data? In *Proceedings of the European Conference on Product and Process Modelling (ECCPM)*, 1995.

[31] J. Bilek and D. Hartmann. Kooperative tragwerksplanung basierend auf multiagentensystemen und adaptiven assoziationsnetzen. In U. Rüppel, editor, *Vernetzt-kooperative Planungsprozesse im Konstruktiven Ingenieurbau*, number 5, chapter Kooperativ, pages 357 – 380. Springer, Darmstadt, Germany, 2007.

[32] BIMServer. `http://bimserver.org`, 26. April 2011.

[33] Stefan Boddy, Yacine Rezgui, Grahame Cooper, and Matthew Wetherill. Computer integrated construction: A review and proposals for future direction. *Advances in Engineering Software*, 38(10):677 – 687, 2007.

[34] buildingSMART. `http://www.buildingsmart.com`, 08. April 2011.

[35] buildingSMART. http://buildingsmart-tech.org/ifc/IFC2x3/TC1/html/index.htm, 08. April 2011.

[36] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. *Pattern-Oriented Software Architecture, Volume 4: A Pattern Language for Distributed Computing*. Wiley, Chichester, UK, 2007.

[37] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. *Pattern-Oriented Software Architecture, Volume 5: On Patterns and Pattern Languages*. Wiley, Chichester, UK, 2007.

[38] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture, Volume1: A System of Patterns*, volume 1. Wiley, Chichester, UK, 1996.

[39] Tomo Cerovsek. A review and outlook for a [']building information model' (bim): A multi-standpoint framework for technological development. *Advanced Engineering Informatics*, In Press, Corrected Proof, 2010.

[40] Po-Han Chen, Lu Cui, Caiyun Wan, Qizhen Yang, Seng Kiong Ting, and Robert L.K. Tiong. Implementation of ifc-based web server for collaborative building design between architects and structural engineers. *Automation in Construction*, 14(1):115 – 128, 2005.

[41] CIMSteel Integration Standards (CIS/2). http://www.cis2.org, 12. April 2011.

[42] James W. Cooper. *Java Design Patterns: A Tutorial*. Addison-Wesley, Reading, MA, 2000.

[43] H.-H. Do, S. Melnik, and E. Rahm. Comparison of schema matching evaluations. In *Proceedings of the 2nd International Workshop on Web Databases (German Informatics Society)*, pages 221 – 237, 2002.

[44] Paul Dyson and Andrew Longshaw. *Architecting Enterprise Solutions: Patterns for High-Capability Internet-based Systems*. Wiley, May 2004.

[45] Charles M. Eastman. A data model analysis of modularity and extensibility in building databases. *Building and Environment*, 27(2):135–148, 1992.

[46] Charles M. Eastman. *Building Product Models: Computer Environments Supporting Design and Construction*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1999.

[47] Chuck Eastman, Paul Teicholz, Rafael Sacks, and Kathleen Liston. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. Wiley Publishing, 2008.

[48] M. El-Saboni, G. Aouad, and A. Sabouni. Electronic communication systems effects on the success of construction projects in united arab emirates. *Advanced Engineering Informatics*, 23(1):130–138, 2009.

[49] A. Halim Elamy. Perspectives in agents-based technology. `http://www.agentlink.org/newsletter/18/AL-18.pdf`, August 2005. AgentLink News - Issue 18.

[50] Eurostep. `http://www.eurostep.com`, 26. April 2011.

[51] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89 – 124, 2005.

[52] I. Faraj and M. Alshawi. Global project collaboration in the construction industry: standardisation and integration. *International Journal of Computer Applications in Technology*, 20(1 - 3):26 – 41, 2004.

[53] I. Faraj, M. Alshawi, G. Aouad, T. Child, and J. Underwood. An industry foundation classes web-based collaborative construction computer environment: Wisper. *Automation in Construction*, 10(1):79 – 99, 2000.

[54] M. Farhi, G. Aouad, and G. Cooper. Osconcad: A model-based cad system integrated with computer applications. *Journal of Information Technology in Construction (ITcon)*, 3(3):26 – 46, 1998.

[55] FIATECH. `http://fiatech.org`, 26. April 2011.

[56] B. Firmenich, C. Koch, T. Richter, A. H. Olivier, and D. G. Beer. Cademia: A platform for the development of civil engineering applications. In *Proceedings of the Twelfth International Conference (ICCCBE-XII)*, 2008.

[57] B. Firmenich and E. Rank, editors. *Überblick zum Themenbereich Verteilte Produktmodelle. In: Abschlussbericht zum DFG-Schwerpunktprogramm 'Vernetzt-kooperative Planungsprozesse im Konstruktiven Ingenieurbau'*. Springer Verlag, Heidelberg, 2007.

[58] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison Wesley, Reading, Massachusetts, 2002.

[59] Bundesanstalt für Wasserbau (BAW). `http://nokis.baw.de`, 26. April 2011.

[60] Norbert Frisch. *Verfahren zur Unterstützung der Arbeitsabläufe bei der Crash-Simulation im Fahrzeugbau*. PhD thesis, Insitut für Visualisierung und Interaktive Systeme, Universität Stuttgart, 2004.

[61] T. Froebel, B. Firmenich, and C. Koch. Coupling patterns in civil engineering applications. In *Proceedings of the 18th International Conference on the Applications of Computer Science and Mathematics in Architecture and Civil Engineering (IKM)*, 2009.

[62] T. Froebel, B. Firmenich, and C. Koch. Formalisation and assessment of data structure based coupling in distributed civil engineering applicatons. In *Proceedings of the ICCCBE*, 2010.

[63] Thomas Froese, Martin Fischer, Francois Grobler, John Ritzenthaler, Kevin Yu, Stuart Sutherland, Sheryl Staub, Burcu Akinci, Ragip Akbas, Bonsang Koo, Alex Barron, and John Kunz. Industry foundation classes for project management - a trial implementation. *Journal of Information Technology in Construction (ITcon)*, 4:17 – 36, 1999.

[64] Thomas Froese, Zonghai Han, and Michael Alldritt. Study of information technology development for the canadian construction industry. *Canadian Journal of Civil Engineering*, 34, 2007.

[65] Thomas M. Froese and Boyd C. Paulson. Opis: An object model–based project information system. *Computer-Aided Civil and Infrastructure Engineering*, 9(1):13 – 28, 1994.

[66] Ariel Fuxman, Mauricio A. Hernandez, Howard Ho, Renee J. Miller, Paolo Papotti, and Lucian Popa. Nested mappings: schema mapping reloaded. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB)*, VLDB '06, pages 67 – 78. VLDB Endowment, 2006.

[67] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1. edition, November 1994.

[68] Andreas Geiger. Produktdatenmodelle im bauwesen - ifc im praxistest -. Master's thesis, Fachhochschule Karlsruhe, Fachbereich Architektur und Bauwesen, 2001.

[69] Wim Gielingh. An assessment of the current state of product data technologies. *Computer-Aided Design*, 40:750–759, 2008.

[70] Augenbroe G.L.M. An overview of the combine project. In *Proceedings of the First European Conference on Product and Process Modelling in the Building Industry (ECPPM)*, pages 547 – 554, 1995.

[71] W3C / Web Services Glossary. `http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211`, 26. April 2011.

[72] CADEMIA-Consult GmbH. `http://www.cademia.org`, 08. April 2011.

[73] Martin Grabmüller. Multiparadigmen-programmiersprachen. Technical report, Technische Universität Berlin, Institut für Softwaretechnik und Theoretische Informatik, Fachgebiet Übersetzerbau und Programmiersprachen, October 2003.

[74] Mark Grand. *Patterns in Java: A Catalog of Reusable Design Patterns Illustrated with UML, Volume 1*. Wiley, New York, 2002.

[75] OMG Object Managment Group. `http://www.corba.org`, 26. April 2011.

[76] OMG Object Managment Group. `http://www.uml.org`, 26. April 2011.

[77] J.C. Grundy, J.G. Hosking, R. Amor, W.B. Mugridge, and Y. Li. Domain-specific visual languages for specifying and generating data mapping systems. *Journal of Visual Languages and Computing*, 15:243 – 263, 2004.

[78] Robert Hanmer. *Patterns for Fault Tolerant Software*. Wiley Publishing, 2007.

[79] Bauinformatik Hannover. `http://nokis.bauinf.uni-hannover.de/www/Projekt.html`, 26. April 2011.

[80] George T. Heineman and William T. Councill. *Component-Based Software Engineering: Putting the Pieces Together (ACM Press)*. Addison-Wesley Professional, 2001.

[81] Mario Höcker and Toni Fröbel. Digitale atlanten - interpolationsverfahren und datenmanagement. In *Forum Bauinformatik - Junge Wissenschaftler forschen*, Weimar, 2006. Verlag der Bauhaus-Universität Weimar.

[82] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[83] Faraj I and Alshawi M. A modularised integrated computer environment for the construction industry: Space. *Journal of Information Technology in Construction (ITcon)*, 4:37 – 52, 1999.

[84] April 2006 IAI Forum Denmark, IFC Exchange Test between 3D CAD applications. `http://cic.vtt.fi/projects/vbe-net/data/Danish_IFC_Exchange_Test_April_2006.pdf`, 26. April 2011.

[85] IEEE - Institute of Electrical and Electronics Engineers. *IEEE Standard 610.12-1990: Glossary of Software Engineering Terminology*, September 1990.

[86] Mark Inverno and Michael Luck. *Understanding agent systems*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[87] U. Isikdag, G. Aouad, J. Underwood, and S. Wu. Building information models: a review on storage and exchange mechanisms. In *Proceedings of the CIB W78's 24th International Conference on IT in Construction*, pages 135–143, Maribor, Slovenia, 2007.

[88] Dimyadi J., Amor R., and Spearpoint M. Sharing building information using the ifc data model for fds fire simulation. Karlsruhe, Germany, 2008. Proceedings of 9th IAFSS Symposium on Fire Safety Science.

[89] Y.-S. Jeong, C.M. Eastman, R. Sacks, and I. Kaner. Benchmark tests for bim data exchanges of precast concrete. *Automation in Construction*, 18(4):469–484, 2009.

[90] Kaj A. Jørgensen, Jørn Skauge, Per Christiansson, Kjeld Svidt, Kristian Birch Sørensen, and John Mitchell. Use of ifc model servers - modelling collaboration possibilities in practice. Technical report, Aalborg University [Department of Production,Department of Civil Engineering], Aarhus School of Architecture [Department of Building Design], May 2008.

[91] Lee J.Y. Shape representation and interoperability for virtual prototyping in a distributed design environment. *The International Journal of Advanced Manufacturing Technology*, 17:425 – 434, 2001.

[92] Bletzinger K.-U. and Lähr A. Prediction of interdisciplinary consequences for decisions in aec design processes. *Journal of Information Technology in Construction (ITcon)*, 11:529 – 545, 2006.

[93] Kaan, Ozcan Can, and Ozen Metin Divringi. Coupling the anybody and ansys software suites for biomedical applications. Technical report, Ozen Engineering, Inc., 2007.

[94] Calvin Kam and Martin Fischer. Product model & 4d cad: Final report. `http://www.stanford.edu/group/4D/download/c1.html`, October 2002.

[95] Vineet R. Kamat and Robert R. Lipman. Evaluation of standard product models for supporting automated erection of structural steelwork. *Automation in Construction*, 16(2):232 – 241, 2007.

[96] Hoonsig Kang and Ghang Lee. Development of an object-relational ifc server. Jeju, Korea, 2009. International Conference on Computer Engineering and Management (ICCEM)/ International Centre for Complex Project Management (ICCPM).

[97] Antti Karola, Hannu Lahtela, Reijo Hänninen, Rob Hitchcock, Qingyan Chen, Stephen Dajka, and Kim Hagström. Bspro com-server–interoperability between software tools using industrial foundation classes. *Energy and Buildings*, 34(9):901 – 907, 2002.

[98] T. Khedro, C.M. Eastman, R. Junge, and T. Liebich. Translation methods for integrated building engineering. In *Proceedings of the ASCE Conference on Computing*, pages 579 – 585, 1996.

[99] Michael Kircher and Prashant Jain. *Pattern-Oriented Software Architecture, Volume 3: Patterns for Resource Management*. Wiley, Chichester, UK, 2004.

[100] Phokion G. Kolaitis. Schema mappings, data exchange, and metadata management. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '05, pages 61–75, New York, NY, USA, 2005. ACM.

[101] Ghang Lee, Jongsung Won, Sungil Ham, and Yuna Shin. Metrics for quantifying the similarities and differences between ifc files. *Journal of Computing in Civil Engineering*, 25:172 – 182, 2011.

[102] Maurizio Lenzerini. Data integration: a theoretical perspective. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 233 – 246, New York, NY, USA, 2002. ACM.

[103] Barbara Staudt Lerner and A. Nico Habermann. Beyond schema evolution to database reorganization. *SIGPLAN Not.*, 25:67–76, September 1990.

[104] R. R. Lipman. Mapping between the cimsteel integration standards (cis/2) and industry foundation classes (ifc) product model for structural steel. In *Proceedings of the International Conference on Computing in Civil and Building Engineering (ICCCBE)*, Montreal, Canada, 2006.

[105] R. R. Lipman. Details of the mapping between the cis/2 and ifc product data models for structural steel. *Journal of Information Technology in Construction (ITcon)*, 14:1–13, 2009.

[106] Robert Lipman, Mark Palmer, and Sebastian Palacios. Assessment of conformance and interoperability testing methods used for construction industry product models. *Automation in Construction*, 20(4):418 – 428, 2011.

[107] Michael Luck, Peter McBurney, Onn Shehory, and Steven Willmott. Challenges for agent-based computing. `http://www.agentlink.org/newsletter/19/AL-19.pdf`, November 2005. AgentLink News - Issue 19.

[108] Dayal M. Analyse des 3d-datenaustausches via ifc-modell am beispiel komplexer objektdokumentation in der automobilindustrie mit dem ziel der optimierung von planungsprozessen. Master's thesis, Technische Universität München, Fakultät für Architektur, Lehrstuhl für Baurealisierung und Bauinformatik, 2004.

[109] Halfawy M. and Froese T. Building integrated architecture/engineering/construction systems using smart objects: a methodology and implementation. *Journal of Computing in Civil Engineering*, 19(2):172 – 181, 2005.

[110] Halfawy M. and Froese T. Component-based framework for implementing integrated architectural/engineering/construction project systems. *Journal of Computing in Civil Engineering*, 21(6):441 – 452, 2007.

[111] H. Ma, K.M.E. Ha, C.K.J. Chung, and R. Amor. Testing semantic interoperability. In *Proceedings of the Joint International Conference on Computing and Decision Making in Civil and Building Engineering (JICCDMCBE)*, Montreal, Canada, 2006.

[112] Jayant Madhavan and Alon Y. Halevy. - composing mappings among data sources. In Johann-Christoph Freytag, Peter Lockemann, Serge Abiteboul, Michael Carey, Patricia Selinger, and Andreas Heuer, editors, *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB)*, pages 572 – 583. Morgan Kaufmann, San Francisco, 2003.

[113] Nour Mohamed Magdy. *A flexible model for incorporating construction product data into building information models*. tech.diss., Bauhaus-Univ. Weimar, Fakultät Bauingenieurwesen, 2006.

[114] SFB524 Materials and Structures in Revitalisation of Buildings. `http://www.uni-weimar.de/sfb/`, 26. April 2011.

[115] D. Mcilroy. Mass-produced software components. In *Proceedings of the 1st International Conference on Software Engineering, Garmisch Pattenkirchen, Germany*, 1968.

[116] Steven J. Metsker and William C. Wake. *Design Patterns in Java*. Addison-Wesley, Upper Saddle River, NJ, 2. edition, 2006.

[117] Steven John Metsker. *Design Patterns in C#*. Addison-Wesley Professional, 2004.

[118] Microsoft. `http://www.microsoft.com`, 26. April 2011.

[119] Peter Milbradt and Catrin Dorow. Identification of morphological tendencies and velocities in coastal zones. In *Science and Information Technologies for Sustainable Management of Aquatic Ecosystems*. Proceedings of HEIC, 2009.

[120] I. Mittrup, K. Smarsly, D. Hartmann, and V. Bettzieche. An agent-based approach to dam monitoring. In *Proceedings of the 20th CIB/W78 Conference on Information Technology in Construction*, pages 239 – 246, 2003.

[121] Moonki, Cho Hyundeok, Roh Taehwan, and Lee Kunwoo Jung. Integrated framework for vehicle interior design using digital human modeling. *Journal of Computer Science and Technology*, 24(6):1149–1161, 2009.

[122] Anita Moum, Christian Koch, and Tore I. Haugen. What did you learn from practice today? exploring experiences from a danish r&d effort in digital construction. *Advanced Engineering Informatics*, 23(3):229 – 242, 2009.

[123] Gallaher M.P., O'Connor A.C., Dettbarn J.L., and Gilday L.T. Cost analysis of inadequate interoperability in the us capital facilities industry. Technical report, US Department of Commerce Technology Administration, National Institute of Standards and Technology (NIST), 2004.

[124] M. Mueller, J. Ruben, and U. Meissner. Dynamically distributed and p-adaptive fe-simulation of soil-structure-interaction based on multi-agent-systems. In *Proceedings*

of the 3rd MIT Conference on Computational Fluid and Solid Mechanics, pages 993 – 997, 2005.

[125] Glenford J Myers. *Reliable software through composite design*. Petrocelli/Charter, 1975.

[126] Hyacinth S. Nwana. Software agents: An overview. *Knowledge Engineering Review*, 11(3):205 – 244, 1996.

[127] William J. O'Brien, Joachim Hammer, Mohsin Siddiqui, and Oguzhan Topsakal. Challenges, approaches and architecture for distributed process integration in heterogeneous environments. *Advanced Engineering Informatics*, 22(1):28 – 44, 2008.

[128] IEEE Institute of Electrical and Electronics Engineers. `http://www.ieee.org`, 26. April 2011.

[129] Oracle. `http://www.oracle.com`, 26. April 2011.

[130] Kolbe P. Ako - arbeitskreis objekte eine schnittstelle für modellverwaltungssysteme. Weimar, Germany, 1998. Proceedings 10th Forum Bauinformatik.

[131] Meilir Page-Jones. *The Practical Guide to Structured Systems Design.* YOURDON Press, 1980.

[132] D.L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15:1053–1058, 1972.

[133] H. Van Dyke Parunak. What can agents do in industry, and why? an overview of industrially-oriented r&d at cec. In *Proceedings of the Second International Workshop on Cooperative Information Agents II, Learning, Mobility and Electronic Commerce for Information Discovery on the Internet*, pages 1 – 18, London, UK, 1998. Springer-Verlag.

[134] T. Pazlar and Z. Turk. Analysis of the geometric data exchange using the ifc. In *Proceedings of EC-PPM*, pages 165–171, Valencia, Spain, 2006.

[135] T. Pazlar and Z. Turk. Interoperability in practice: Geometric data exchange using the ifc standard. *Journal of Information Technology in Construction (ITcon)*, 13:362–380, 2008.

[136] Katranuschkov Peter. *A Mapping Language for Concurrent Engineering Processes*. PhD thesis, TU Dresden, Dresden, Germany, 2000.

[137] Jim Plume and John Mitchell. Collaborative design using a shared ifc building model–learning from experience. *Automation in Construction*, 16(1):28 – 36, 2007.

[138] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Higher Education, 7th edition, April 2009.

[139] COMBINE Project. `http://www.opengroup.org/combine/overview.htm`, 26. April 2011.

[140] Amor R. and Anumba C.J. A survey and analysis of integrated project databases. In *Proceedings of Concurrent Engineering in Construction'99*, pages 217 – 228, Espoo, Finland, August 1999. Kluwer, B.V.

[141] Amor R. and Ma H. Preservation of meaning in mapped ifcs. In *Proceedings of EC-PPM*, pages 233–236, Valencia, Spain, 2006.

[142] Amor R., Clift M., Scherer R., Katranuschkov P., Turk Z., and Hannus M. A framework for concurrent engineering — tocee. pages 15 – 22. European Conference on Product Data Technology, April 1997.

[143] Greening R. and Edwards M. Atlas implementation scenario. In *Proceedings of the First European Conference on Product and Process Modelling in the Building Industry (ECPPM)*, pages 467 – 472, 1995.

[144] Hitchcock R. Acquiring geometry from cad for energyplus through the use of industry foundation classes. *Building Energy Simulation User News*, 21(5):2 – 5, 2000.

[145] Scherer R. Eu-project combi — objectives and overview. In *Proceedings of the First European Conference on Product and Process Modelling in the Building Industry (ECPPM)*, pages 503 – 510, 1995.

[146] E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *The International Journal on Very Large Data Bases (VLDB)*, 10:334 – 350, 2001.

[147] Alexander Rensch. Kopplung von fem-programmen am beispiel von sysweld und ansys. Master's thesis, Bauhaus-Universität Weimar, 2009.

[148] Luis Reynoso, Esperanza Manso, Marcela Genero, and Mario Piattini. Assessing the influence of import-coupling on ocl expression maintainability: A cognitive theory-based perspective. *Inf. Sci.*, 180:3837 – 3862, October 2010.

[149] Y Rezgui and A Zarli. Paving the way to the vision of digital construction: a strategic roadmap. *Journal of Construction Engineering and Management*, 132(7):767–776, July 2006.

[150] Linda Rising. *The Pattern Almanac*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[151] Roland and Hager Josef Marzy, editors. *Kopplung von 1D- und 3D-Simulationsmethoden zur Optimierung von Kfz-Kühlsystemen*, ÖIAV Tagung. Steyr-Daimler-Puch, Engineering Center Steyr, 2000.

[152] Uwe Rüppel, editor. *Vernetzt-kooperative Planungsprozesse im konstruktiven Ingenieurbau : Grundlagen, Methoden, Anwendung und Perspektiven zur vernetzten Ingenieurkooperation*. Springer, Berlin u.a., 2007.

[153] Alda S., Bilek J., Cremers A. B., and Hartmann D. Awareness and workflow based coordination of networked co-operations in structural design. *Journal of Information Technology in Construction (ITcon), Special Issue Process Modelling, Process Management and Collaboration*, 11:469 – 507, 2006.

[154] Hans Schevers, John Mitchell, Paul Akhurst, David Marchant, Stuart Bull, Kevin McDonald, Robin Drogemuller, and Chris Linning. Towards digital facility modelling for sydney opera house using ifc and semantic web technology. *Journal of Information Technology in Construction*, 12:347 – 362, 2007.

[155] Douglas C. Schmidt, Michael Stal, Hans Rohnert, and Frank Buschmann. *Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects*. Wiley, Chichester, UK, 2000.

[156] Markus Schumacher, Eduardo Fernandez-Buglioni, Duane Hybertson, Frank Buschmann, and Peter Sommerlad. *Security Patterns : Integrating Security and Systems Engineering*. John Wiley & Sons, 2006.

[157] Till Schümmer and Stephan Lukosch. *Patterns for Computer-Mediated Interaction*. Wiley & Sons, Chichester, UK, 2007.

[158] Weiming Shen, Qi Hao, Helium Mak, Joseph Neelamkavil, Helen Xie, John Dickinson, Russ Thomas, Ajit Pardasani, and Henry Xue. Systems integration and collaboration in architecture, engineering, construction, and facilities management: A review. *Advanced Engineering Informatics*, 24:196–207, April 2010.

[159] Weiming Shen, Qi Hao, and Yunjiao Xue. An agent-based service-oriented approach for facility lifecycle information integration and decision supports. In *Proceedings of the IEEE International Conference on Systems Man and Cybernetics (SMC)*, pages 1945 – 1952, 2010.

[160] Abraham Silberschatz, Henry Korth, and S. Sudarshan. *Database Systems Concepts*. McGraw-Hill, Inc., New York, NY, USA, 5 edition, 2006.

[161] M. Sun, G. Aouad, N. Bakis, S. Birchall, and W. Swan. Gallicon - a prototype for the design of water treatment plants using an integrated project database. *International Journal of Computer-integrated Design and Construction*, 2(3):157 – 165, 2000.

[162] Jotne EPM Technology. `http://www.epmtech.jotne.com`, 26. April 2011.

[163] W. B. Teeuw, J. R. Liefting, R. H. J. Demkes, and M. A. W. Houtsma. Experiences with product data interchange: On product models, integration, and standardisation. *Computers in Industry*, 31(3):205–221, 1996.

[164] M. Theiss. *Agentenbasierter Modellverbund am Beispiel des baulichen Brandschutzes in der Gebäudeplanung.* PhD thesis, Technical University of Darmstadt, nstitute of Numerical Methods and Informatics in Civil Engineering, Germany, 2005.

[165] Jan Tulke. *Kollaborative Terminplanung auf Basis von Bauwerksinformationsmodellen*. tech.diss., Bauhaus-Univ. Weimar, Fakultät Bauingenieurwesen, Professur Informatik im Bauwesen, 2010.

[166] Z. Turk. *Internet information and communication systems for civil engineering: a review*, pages 1 – 26. Saxe-Coburg Publications, 2001.

[167] Amar V., Zarli A., Debras P., and Poyet P. Distributing step models with corba. *International Symposium on Global Engineering Networking (GEN'97)*, HNI-Verlagsschriftenreihe(21):79 – 96, 1997.

[168] Renaud Vanlande, Christophe Cruz, and Christophe Nicolle. Managing ifc for civil engineering projects. In *Proceedings of the twelfth international conference on Information and knowledge management*, CIKM '03, pages 179 – 181, New York, NY, USA, 2003. ACM.

[169] Renaud Vanlande, Christophe Nicolle, and Christophe Cruz. Ifc and building lifecycle management. *Automation in Construction*, 18(1):70 – 78, 2008.

[170] Nemetschek Vectorworks. `http://www.nemetschek.com`, 26. April 2011.

[171] J.S.M Vergeest and I. Horváth. Where interoperability ends. In *Proceedings of the ASME Design Engineering Technical Conference (DETC)*, 2001.

[172] M. Verhoef, T. Liebich, and R. Amor. A multi-paradigm mapping method survey. In *Proceedings of the CIB W78 – TG10 Workshop on Modeling of Buildings through their Life-cycle*, pages 233 – 247, 1995.

[173] Markus Völter, Michael Kircher, and Uwe Zdun. *Remoting Patterns - Foundations of Enterprise, Internet, and Realtime Distributed Object Middleware*. J. Wiley & Sons, Hoboken, NJ, USA, October 2004.

[174] Markus Völter, Alexander Schmid, and Eberhard Wolff. *Server Component Patterns: Component Infrastructures Illustrated with EJB*. John Wiley & Sons, Inc, New York, NY, USA, 2002.

[175] David W., Divringi Kaan, and Ozcan Can Wagner. Internal forces of the femur: An automated procedure for applying boundary conditions obtained from inverse dynamic analysis to finite element simulations. Technical report, Ozen Ingenieering, Inc., xxxx.

[176] The World Wide Web Consortium (W3C). `http://www.w3.org/`, 26. April 2011.

[177] W3C / Semantic Web. `http://www.w3.org/standards/semanticweb/`, 26. April 2011.

[178] W3C / Semantic Web. `http://www.w3.org/2001/sw/`, 26. April 2011.

[179] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115 – 152, 1995.

[180] Rezgui Y. and Cooper G. A proposed open infrastructure for construction project document sharing. *Journal of Information Technology in Construction (ITcon)*, 3:11 – 25, 1998.

[181] Song Y., Hamilton A., and Wang H. Built environment data integration using nd modeling. *Journal of Information Technology in Construction (ITcon)*, 12:429 – 442, 2007.

[182] QZ. Yang. Ifc-compliant design information modelling and sharing. *Journal of Information Technology in Construction (ITcon)*, 8:1 – 14, 2003.

[183] Q.Z. Yang and Y. Zhang. Semantic interoperability in building design: Methods and tools. *Computer-Aided Design*, 38(10):1099 – 1112, 2006.

[184] Kevin Yu, Thomas Froese, and Francois Grobler. A development framework for data models for computer-integrated facilities management. 9(2):145 – 167, 2000.

[185] A. Zarli, V. Amar, F. Diard, M. Marache, and P. Poyet. Bridging the gap between step,corba and virtual reality technology for the next building industry applications generation. In *Proceedings of the 4th International Conference on Concurrent Enterprising (ICE'97)*, pages 219 – 229, Nottingham, UK, October 1997.

[186] A. Zarli and P. Poyet. A framework for distributed information management in the virtual enterprise: The vega project. In *Proceedings of the IFIP TC5 WG5.3 / PRODNET Working Conference on Infrastructures for Virtual Enterprises: Networking Industrial Enterprises*, pages 293 – 306, Deventer, The Netherlands, 1999. Kluwer, B.V.

[187] Alain Zarli. Integrating step and corba for applications interoperability in the future virtual enterprises computer-based infrastructures. In *Proceedings of the 1997 IASTED International Conference on Intelligent Information Systems (IIS '97)*, pages 309 –, Washington, DC, USA, 1997. IEEE Computer Society.

[188] Roberto Zicari. A framework for schema updates in an object-oriented database system. In *Proceedings of the Seventh International Conference on Data Engineering*, pages 2–13, Washington, DC, USA, 1991. IEEE Computer Society.